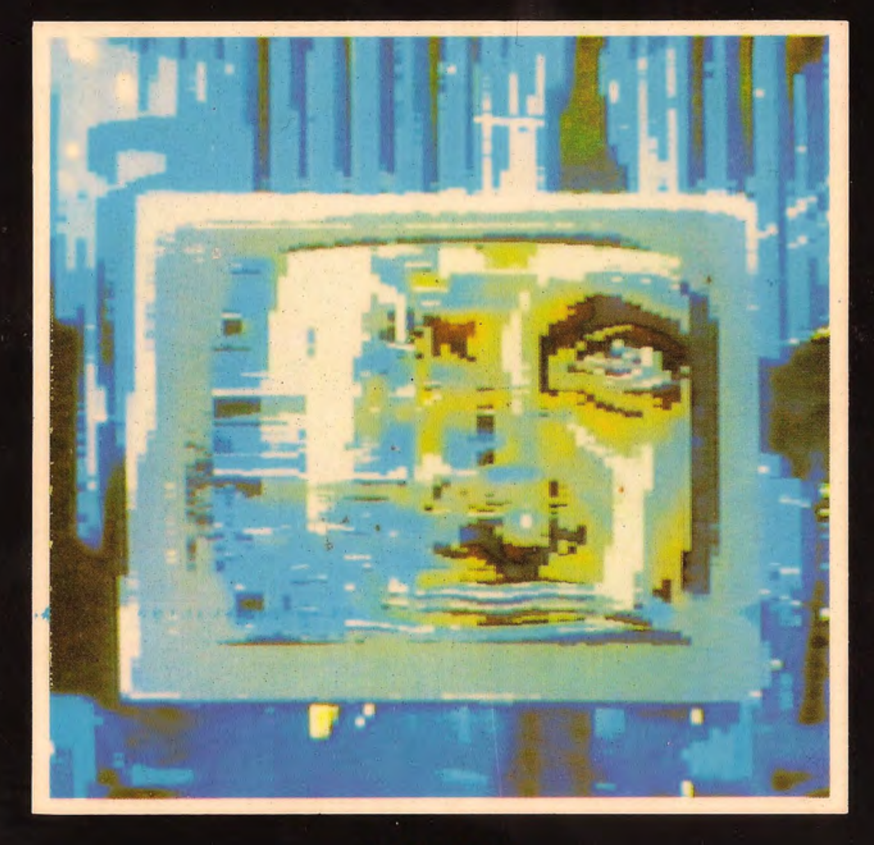
# BIBLIOTECA BASICA
# INFORMATICA

## DENTRO Y FUERA DEL ORDENADOR 1



INGELEK

# BIBLIOTECA BASICA INFORMATICA

## DENTRO Y FUERA DEL ORDENADOR 1

INGELEK

ISBN del tomo: 84-85831-32-2
ISBN de la obra: 84-85831-31-4
Fotocomposición: Pérez Díaz, S. A.
Imprime: Héroes, S. A. - 28016 Madrid.
Depósito Legal: M-36.687-1985

# INDICE

# PROLOGO

LA rapidísima evolución de la microelectrónica ha hecho posible que la adquisición de un ordenador personal sea algo prácticamente al alcance de cualquiera. Y no son juguetes precisamente: la mayoría están dotados de una potencia de cálculo igual o superior a la de un enorme ordenador de los años 50 que, no obstante, a duras penas cabía en una sola habitación. ¡Y usted lo puede colocar sobre su escritorio!

Pero la revolución tecnológica no ha influido únicamente en las dimensiones de la máquina. En efecto, basta hojear cualquier revista especializada para verse "asaltado" por páginas y páginas con publicidad de ordenadores, accesorios, periféricos cada vez más perfeccionados y, por supuesto, montañas de software. En esta verdadera jungla de opciones, el usuario (o el que piense serlo en el futuro) puede perderse con facilidad, si no tiene un mínimo de conocimientos sobre lo que contienen en realidad esas cajas tan bien presentadas. Es necesario saber qué hay dentro y fuera de ése que puede ser un ordenador. Algo así ocurre con aquellos automóviles que tienen carrocerías muy parecidas, pero cuyos motores y mecánica son muy diferentes. No debemos dejarnos engañar.

El conjunto de elementos físicos que constituyen un ordenador se conoce como su hardware, mientras que el soft-

ware son la serie de programas que le permiten funcionar del modo correcto. Ahora bien, ambas partes son interdependientes: si el hardware es poco potente, será difícil obtener el máximo rendimiento incluso del mejor software disponible y viceversa. Por ello, una de las intenciones de este libro es precisamente introducir al lector en el mundo de los chips, que constituyen el "corazón" del sistema (CPU, RAM, ROM, etc.) y al de aquellos que van "por fuera" (periféricos).

Conocer el hardware ayuda mucho a la hora de seleccionar un ordenador o, si ya posee uno, le permitirá sacarle al máximo provecho y tener sólidos criterios para la posterior adquisición de impresoras, memorias de masa y periféricos diversos. Además ¿cómo es posible "convivir" con nuestro propio ordenador personal sin tener siquiera el más pequeño deseo de curiosear en su interior?

Si hemos logrado transmitirle "el gusanillo" acompáñenos en nuestro viaje: trataremos de adentrarnos juntos en el mundo del hardware: tarjetas principales, de expansión, de aplicación, de comunicaciones... y ¡cómo no! veremos los chips, esos circuitos electrónicos, integrados en una pastilla, que cubren la superficie de cualquier tarjeta y que familiarmente se denominan "cucarachas".

Nuestra forma de exponerle los temas será seria y clara, sin resultar pesada (esperamos), con una progresiva profundización para evitar sustos. Con esta misma finalidad hemos incorporado a cada capítulo un glosario, concebido para ayudarle en la comprensión de los términos más áridos o menos conocidos, y al que en muchos casos podrá acudir cuando aparezca en el texto algún concepto nuevo para usted.

Para concluir, echemos un vistazo al contenido de los diversos capítulos. Comenzaremos por la apertura del propio ordenador personal, operación imprescindible si queremos examinar lo que se observa a primera vista (tarjetas, elementos mecánicos, disposición de los diversos componentes). En el segundo capítulo hablaremos de los circuitos digitales elementales que, integrados a millares, darán lugar al μP ("microprocesador") y a sus diversos chips de apoyo; así como del funcionamiento de un microprocesador. En el tercer capítulo veremos qué es y cuál es la misión de cada chip asociado a la CPU; estableceremos contacto con las

memorias RAM y ROM, los dispositivos de entrada/salida y sus características principales. En el capítulo cuarto hablaremos del software, considerado como elemento aglutinante y controlador del hardware antes descrito, con lo que llegaremos a la mitad de la exposición. En el capítulo quinto examinaremos, de la forma más amena posible, los "ceros", "unos", códigos hexadecimales, código ASCII, etc., con la finalidad de poder profundizar un nivel más en el hardware. Los dos capítulos siguientes, sexto y séptimo, son los más largos y descriptivos; en ellos conoceremos y analizaremos todos los tipos de periféricos de un ordenador personal, puesto que, llegados a este punto, dispondremos de todos los instrumentos necesarios para hacerlo con la máxima objetividad. El capítulo octavo, a modo de conclusión, trata de fijar las ideas y dar algunas recomendaciones finales al lector.

Confiamos en que nuestro empeño de hacer asimilables unas ideas y conceptos tan áridos no nos lleve a perder esa rigurosidad también necesaria. Usted, amigo lector, será el juez.

## *Algunas consideraciones sobre el aspecto externo antes de adentrarnos en el interior*

En la fotografía 1 se muestran algunos de los ordenadores personales más difundidos. Quizás el suyo se encuentre entre ellos. De inmediato observará una característica común en su estructura externa: todos tienen una caja que contiene la parte principal del sistema electrónico y que, en algunos modelos, incluye también el teclado (Apple, Commodore, Spectrum,...) y/o las unidades de disco flexible (IBM-PC). Asimismo, también hay un aparato con forma de pantalla de televisión, que recibe el nombre de monitor de vídeo. En su pantalla podremos visualizar caracteres alfanuméricos (letras y números) o gráficos. En la mayor parte de los casos el monitor es exterior y va separado de la estructura principal, aunque hay excepciones que lo incorporan en la propia caja. En los ordenadores personales, denominados "portátiles" (por sus reducidas dimensiones) es co-



*Foto 1 - Estos son algunos de los ordenadores personales más conocidos.*

mún sustituir el monitor por un visualizador ("display") de cristal líquido.

La elección de un ordenador personal efectuada sobre la base de consideraciones exclusivamente estéticas no suele ser conveniente, aunque desde luego su presentación sea un detalle a tener en cuenta. Las dimensiones, peso, facilidad de manejo (ergonomicidad) portabilidad y durabilidad de su teclado influyen, a veces de forma determinante, en la buena relación entre usuario y sistema. Al abrir la tapa, podremos también valorar la mecánica, distribución y orden del conjunto y esto es importante no porque tengamos que curiosear su interior como actividad cotidiana, sino más bien porque se debe evaluar por anticipado los problemas que presentarán el mantenimiento e incluso la reparación.

Por contra, desde el exterior se puede valorar mejor la robustez del aparato. Antes que cajas de chapa son preferibles materiales plásticos resistentes a los golpes, y mucho mejor si disponen de refuerzos internos metálicos. Así tendremos cajas más ligeras, resistentes y, normalmente, más estéticas. Una buena caja asegura una vida más larga a los componentes internos.

En las fotografías 2 y 3 aparece, a título de ejemplo, el famoso Apple II (en la versión "e"). En la primera vemos su aspecto externo, mientras que la segunda nos permite observar con claridad su interior, donde pueden verse la tarjeta principal (única) que sirve de soporte a toda la parte electrónica y la fuente de alimentación, protegida por una caja metálica.

En un buen ordenador personal, la estructura interna debe ser sencilla y cada componente tener un fácil acceso. Por ello, quizá valdría la pena abrir la famosa tapa en la misma tienda, antes de adquirirlo. En cualquier caso, es fácil indicar algunas normas que sirvan de base en la evaluación. Ante todo, cuantos menos componentes haya, menores serán las posibilidades de avería. En las máquinas de reciente factura, un número reducido de componentes indica que se han empleado circuitos integrados "hechos a medida" ("custom IC"), por lo que cabe pensar que se ajustará mejor a lo que se espera de ella y que, desde el punto de vista tecnológico, será más moderna.

En segundo lugar debemos valorar el diseño y realiza-

*Foto 2 - Unidad principal del Apple II-e.*

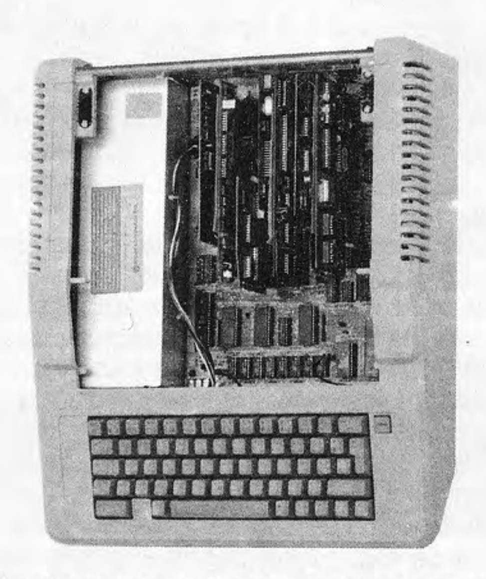


*Foto 3 - Interior del Apple II, con la tarjeta principal y la caja de la fuente de alimentación visibles.*

ción de las placas electrónicas existentes. El empleo de zócalos (naturalmente si son de calidad) es una ventaja, pues facilita enormemente las tareas de mantenimiento y reparación, en contraposición con aquellas placas en las que todos los componentes están soldados. Si existen conectores, deberán ser fácilmente accesibles, tener una conexión robusta y segura y, a ser posible, normalizadas (estándar), pues así será más fácil encontrarlos si alguna vez los precisamos. El interior debe estar bien ventilado; la presencia de un ventilador (silencioso, claro) es, sin duda, garantía de una vida más prolongada para los componentes.

Hechas estas consideraciones, podemos centrarnos en la electrónica contenida en el interior, en la tarjeta principal.

## *Arquitectura hardware de un ordenador personal. Bloques funcionales*

La tarjeta principal, a veces la única existente y necesaria para el funcionamiento completo de la máquina, suele mostrar una disposición de componentes que indica la diversidad de funciones de los chips utilizados. Ante todo, el "cabeza de familia" esto es, el microprocesador, siempre será fácilmente identificable: es un circuito integrado grande, con una cápsula que suele tener 40 patillas, como es el caso del 6502 (microprocesador del Apple) o el 8088 (microprocesador del IBM-PC). En las máquinas más modernas, tales como el "Mac" de Apple, el $\mu$P está contenido en una cápsula de 64 patillas; se trata del 68000, mucho más potente que los otros dos, y que, en consecuencia, precisa más patillas.

Más adelante veremos el funcionamiento y cómo está construido un microprocesador. Por ahora, baste decir que el microprocesador recibe también la denominación de "Unidad Central de Proceso" (o CPU) porque su misión es dirigir la ejecución de los programas y el proceso de los datos introducidos por el usuario.

Alrededor de la CPU se encuentran otros circuitos integrados, que suelen ser más pequeños. Una buena parte de ellos son de memoria, actuando, en relación con la CPU, como una especie de "cuarto trastero" para los datos y las órdenes en curso de ejecución. Finalmente, otros circuitos

permiten conectar la tarjeta principal con otras tarjetas exteriores, son los conocidos como "integrados de entrada/salida"; controlan la entrada (y/o salida) de datos hacia (o desde) la tarjeta principal. En posteriores capítulos veremos con detalle las funciones de dichos componentes y también cuáles son los dispositivos que se les pueden conectar.

A modo de "aglutinante" que permite unir los tres tipos de circuitos integrados antes citados existe en la tarjeta una amplia gama de otros integrados más pequeños, cuya función es la de adaptar las señales que fluyen entre los circuitos integrados principales, facilitando así su conexión.

La necesidad de estos pequeños circuitos auxiliares deriva de que tanto el microprocesador, la memoria y los dispositivos de entrada/salida están diseñados de forma que puedan dar lugar a configuraciones muy distintas en sus necesidades y diseño.

Esa deseada versatilidad se paga al tener que añadir en cada caso concreto un cierto número de estos componentes auxiliares.

Es el momento de tomar una hoja de papel e intentar dibujar la disposición de la tarjeta principal, agrupando las funciones existentes en bloques lógicos. Así obtendremos, no el esquema eléctrico (que es algo siempre bastante complejo, variable de una máquina a otra, y que no entra dentro de los objetivos de este libro), sino un diagrama de bloques funcional de la parte electrónica.

De este modo llegaremos a lo que se ilustra en la figura 1. En un rectángulo que señala los límites de la placa prin-

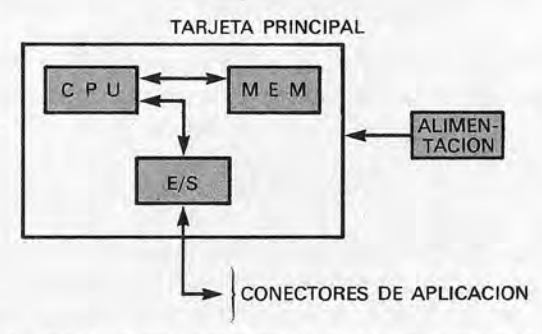



*Fig. 1 - Bloques que constituyen la tarjeta principal del ordenador personal.*

cipal, están contenidos otros tres rectángulos que representan a cada una de las funciones antes citadas: MEM identifica la memoria y está conectada a la CPU; E/S significa entrada/salida y también está conectada a la CPU. Las conexiones son bidireccionales, lo cual quiere decir que el intercambio de información puede producirse en ambos sentidos: desde y hacia la CPU. Por supuesto, una fuente de alimentación suministra la energía necesaria al sistema. Observe que no aparece ninguna referencia al soporte lógico y ello se debe a que, al menos aquí, no resulta necesario ni sirve para nada en particular.

Con el siguiente paso llegamos a la figura 2; en ella la visión es mucho más amplia; la tarjeta principal muestra con más detalle su contenido aunque, naturalmente, todavía no sabemos cuál es. Aparecen tres unidades exteriores a la tarjeta, que en seguida identificaremos por muy poco que hayamos trabajado con nuestro ordenador personal: abajo tenemos un teclado, a la derecha una unidad de presentación visual y a la izquierda una memoria de masa, indicada de forma simbólica por una unidad de disco flexible.

El teclado es un accesorio exterior muy importante, ya que, en principio, es el único medio que tiene el usuario para comunicarse con la tarjeta principal proporcionándole datos y órdenes.

En la figura 2 vemos que el sentido de la conexión va desde el teclado a la tarjeta principal. Conclusión: el tecla-

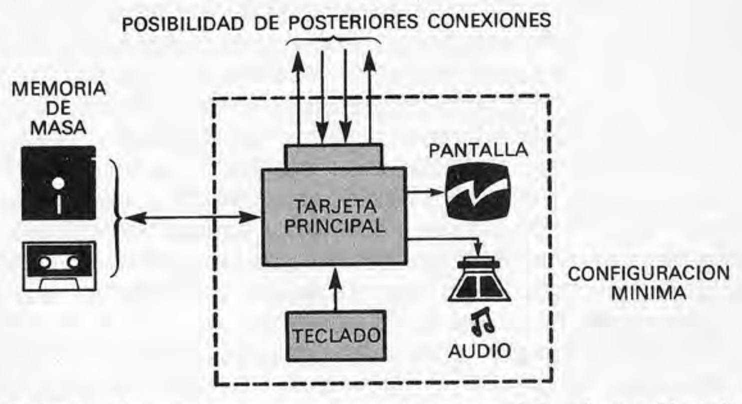



*Fig. 2 - Estructura de un ordenador personal (configuración mínima y periféricos de memoria de masa).*

do es un "periférico" y, más concretamente, un periférico de entrada para el $\mu$P. Por el contrario, la pantalla sirve al ordenador para visualizar las respuestas a nuestras solicitudes en forma "inteligible" por el usuario, mediante frases o figuras preprogramadas en la memoria del propio ordenador.

Así, por ejemplo, si pulsamos PRONT en lugar de PRINT obtendremos la respuesta: SYNTAX ERROR. La unidad de presentación visual es, pues, un periférico exclusivamente de salida y el sentido de conexión va desde la tarjeta principal hacia la propia unidad. Con frecuencia existe una unidad de audio, reducida a su mínima expresión, pero suficiente para conseguir al menos un pitido ("beep") que pueda avisar al programador, acaso adormilado sobre el teclado a las tres de la madrugada. Es otro caso de periférico de salida.

El teclado, el monitor y el dispositivo de audio, junto con la tarjeta principal, consituyen lo que se suele definir como "configuración mínima" de un ordenador personal. También en la figura 2 observamos una línea de trazos que rodea a estos elementos fundamentales, con lo que se indica de forma simbólica la caja que los encierra. La unidad de memoria de masa es un periférico importante. Sin embargo, y sobre todo para no aumentar el precio de la configuración mínima, se suele ofrecer como un accesorio adicional. A pesar de esto, ordenadores como el Apple II-c, el Macintosh, o el IBM-PC pueden tener todos ellos al menos una unidad de disco flexible incorporada en la estructura de base. La memoria de masa, como su nombre indica, permite almacenar de modo permanente grandes cantidades de datos, que sería poco adecuado y demasiado costoso mantener en la memoria interna alojada, como hemos visto, en la tarjeta principal. El almacenamiento se produce, desde el punto de vista conceptual, de la misma forma que cuando grabamos en una cinta de casete una canción. De hecho, en los ordenadores personales más económicos se utilizan soportes magnéticos que son precisamente cintas de casete. Más adelante, en el capítulo dedicado a estos periféricos, analizaremos a fondo su funcionamiento.

Completando el examen de la figura 2, hemos indicado la existencia de una zona dedicada a posteriores conexiones y que se suele denominar "back-panel". En realidad, éste panel contiene los conectores de entrada, de salida y de entrada/salida disponibles para futuras ampliaciones del ordenador personal. En la realización de las conexiones necesarias pueden intervenir otras unidades periféricas e, incluso, otros ordenadores, de modo que se constituya una red de microprocesadores intercomunicados.

Intencionadamente hemos indicado los diversos tipos de conexión posibles: unidireccionales y bidireccionales, útiles cada uno para determinadas clases de periféricos. Todas estas conexiones, incluidas las que afectan a las unidades descritas poco antes, están controladas por circuitos especializados de entrada/salida existentes en la tarjeta principal, lo cual hace más importante aún considerar la potencia y la versatilidad de esta parte del hardware antes de la adquisición, pues un ordenador personal con escasos circuitos de entrada/salida no permitirá un control ágil de los periféricos que se le conecten, y limitará el número posible de estos.

## *Personalización del hardware (las expansiones)*

Muchos y famosos ordenadores personales poseen una tarjeta principal preparada para admitir hardware adicional. Este se presenta bajo la forma de tarjetas que se insertan en los conectores adecuados (los famosos "slots") de la tarjeta principal. Si es este el sistema empleado suelen ser menos los dispositivos de entrada/salida incluidos en la configuración básica; a menudo quedan limitados al dispositivo de audio y a la conexión para la unidad de cinta de casete. Aunque pueda parecer un sistema restrictivo, suele ser por el contrario, una buena opción por parte del fabricante. De ese modo, la tarjeta principal y, por consiguiente, el ordenador personal en la configuración base, cuesta menos y el usuario no se ve obligado a pagar también por lo que no necesita. Así ha nacido un floreciente mercado de pequeñas tarjetas de ampliación que entran, por derecho propio, dentro del concepto de hardware de un ordenador personal.

Naturalmente, entre tantos modelos disponibles, es preciso orientarse con buen criterio para no correr el riesgo de adquirir un producto que no sea verdaderamente compatible con el resto de los circuitos de la tarjeta principal. A tal

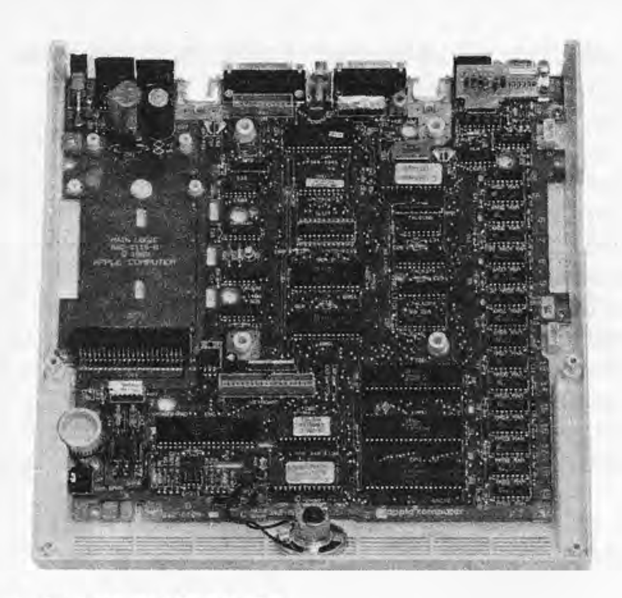



*Foto 4 - Interior del Apple II-c.*

fin, recuerde que la inserción de una tarjeta en una ranura equivale a ampliar, desde el punto de vista electrónico, la tarjeta principal; lo que es casi como realizar modificaciones al circuito de esta última. Por este motivo corremos el riesgo de deteriorar el núcleo de la configuración principal, si insertamos una tarjeta no adecuada.

Dejando aparte algunas tarjetas "piratas", es conveniente dirigirse a los proveedores reconocidos por el fabricante cuando no es este último el que pone a la venta dichas tarjetas adicionales. ¿Cuáles de ellas son verdaderamente necesarias? La respuesta depende exclusivamente de nuestras previsiones de trabajo: existen tarjetas de E/S en paralelo y en serie (veremos más adelante lo que ello significa), para conectar una impresora (quizá el primer periférico en importancia, si damos por descontada la presencia de una memoria de masa), tarjetas de control de la unidad de presentación visual (las de gráficos de alta resolución y las de color están ya a la orden del día); tarjetas de control para todas clases de memorias de masa (discos flexibles y discos rígidos) en los diversos formatos existentes; tarjetas para comunicaciones, para control de señales,... Las reglas de oro



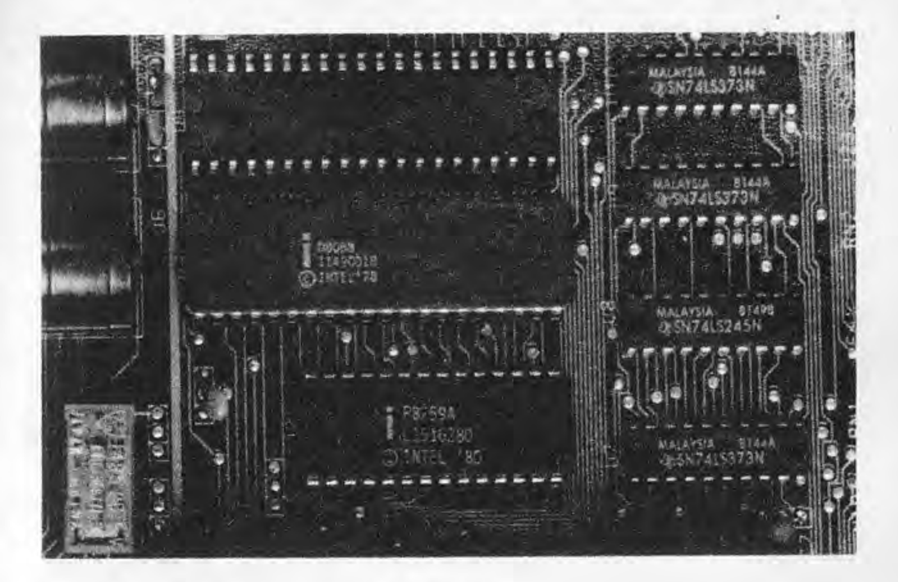


*Foto 5 - Placa principal del IBM-PC.*

aplicables siempre que se adquiera una tarjeta de ampliación son las siguientes: 1) la tarjeta debe ser original, para asegurar la existencia de componentes de calidad; 2) no debe plantear problemas en su control y, en la práctica, ha de funcionar de forma inmediata una vez insertada en su ranura (slot), sin que se tengan que efectuar cada vez que se conecte el ordenador acciones complicadas para su activación y 3) no debe absorber una potencia superior a la que ponga a su disposición la fuente de alimentación del sistema.

Esta última regla es importante porque fácilmente se corre el riesgo de sobrecargar la alimentación del propio ordenador personal (puesto que en las tarjetas adicionales también hay un número considerable de integrados que consumen corriente), que se encontrará entonces, con dificultades. Si en estas condiciones se produce también un pequeño fallo (caída) en la red, todo el trabajo realizado hasta el momento se perderá pues la alimentación no logrará suministrar la energía necesaria. Aparte de estas normas obligadas con respecto al hardware, el usuario podrá elegir las opciones adicionales que desee, aunque las más comunes son: memoria suplementaria e interfaces (sistemas de inter-



Foto 6 - Típicos ordenadores personales portátiles.

conexión) para impresora, vídeo y unidad de disco (flexible o rígido).

Otros ordenadores, que podríamos llamar "cerrados", no tienen ranuras (slots) en la tarjeta principal sino que incluyen en ella todos los interfaces clásicos citados. En tal caso, la tarjeta principal costará más, pero el hecho de haber prescindido de conectores y de espacio para las tarjetas adicionales proporcionará un conjunto más compacto, y a menudo, el coste total del ordenador personal será menor. Este es el caso de las opciones portátiles. En la fotografía 4 se muestra el aspecto interior del Apple II-c, en la 5 el de un IBM-PC, ordenadores personales clásicos, y en la fotografía 6 aparecen algunos ordenadores personales portátiles.



## *Hacia el estudio de los chips*

Bueno, hasta ahora no hemos exigido grandes esfuerzos; nos hemos limitado a abrir el ordenador personal y a explicar su contenido a grandes rasgos.

LLegados a este punto, podemos "visualizar" la arquitectura como un diagrama de bloques formado por los diversos componentes que constituyen el hardware y tenemos también algunos criterios de elección esenciales, tales como la robustez, la facilidad de manejo, durabilidad, fácil mantenimiento y capacidad de ampliación.

Ahora bien, para adentrarse de un modo más profundo en el hardware es necesario conocer los conceptos funcionales de un ordenador, partiendo de aquellos que sirvieron de base en el diseño de los propios circuitos integrados. Veremos luego como la teoría digital es la base de todos los chips presentes en un ordenador personal, así como del funcionamiento de la CPU y la forma en que logra "animar" a la tarjeta principal para que esta, a su vez, "de vida" a nuestro ordenador personal.

## *GLOSARIO*

SISTEMA: Se indica con este término genérico la configuración constituida por el ordenador personal con sus periféricos conectados (impresoras, discos, etc).

DISQUETE (DISCO FLEXIBLE): medio de almacenamiento masivo permanente, constituido por un disco de material magnetizable protegido dentro de una funda de plástico. Una hendidura en dicha funda protectora permite que una cabeza de grabación introduzca los datos en el disco. El disco flexible, universalmente denominado "floppy" ("blando"), necesita de un dispositivo, conocido como "unidad de disco flexible", en donde se inserta para poder ser objeto de lectura o escritura por parte del ordenador.

MONITOR: Es el terminal de imagen usual del ordenador. En una presentación que simula la pantalla de un televisor se visualizan nombres, datos, ódenes y gráficos, tal como se introducen desde el teclado por parte del

usuario, así como las respuestas que proporciona el ordenador.

CARACTER ALFANUMERICO: Se trata de una letra de la A a la Z, un número o bien, un carácter simbólico como: *,.:;$%&, etc.

VISUALIZADOR ("DISPLAY") DE CRISTAL LIQUIDO: Panel plano y delgado en el que la pantalla de presentación visual (es decir, el tubo de rayos catódicos) está sustituido por una placa de cristal líquido. La ventaja (frente a un coste bastante más elevado) es el consumo de corriente, prácticamente nulo, que lo hace aconsejable para los ordenadores portátiles.

PERIFERICO: Se trata de un subsistema, habitualmente controlado por microprocesador, que se conecta al ordenador personal pero que queda físicamente separado. Un ejemplo clásico es la impresora.

SLOT (RANURA): Conector situado en la tarjeta principal del ordenador personal en el que se puede insertar una tarjeta de hardware adicional, también denominada de "expansión".

BACK-PANEL: Panel posterior del ordenador personal, que sirve de receptáculo a los cables y a las conexiones desde y hacia los periféricos. Los conectores que aparecen en el "back-panel" (panel posterior) están conectados, en el interior del ordenador personal, con los circuitos de E/S (entrada/salida) que forman parte integrante de la tarjeta principal o de las ocasionales tarjetas de expansión.



# CAPITULO II

## ESTRUCTURA Y FUNCIONAMIENTO DE LA CPU



AMIGO lector, no se asuste al leer el título de los dos primeros apartados; nuestra intención no es la de abrumarle con toda la historia de la microelectrónica, desde la invención del transistor hasta el microprocesador, sino más bien llegar a comprender cómo millares de transistores, integrados en un solo chip de silicio, pueden cooperar en el establecimiento de las características de potencia y cálculo de una CPU, "corazón" de nuestro ordenador personal.

### Fundamentos de la electrónica digital

Un transistor es en esencia un interruptor controlado eléctricamente. En la figura 1a. se muestra un circuito elemental con un solo transistor que, en la figura 1b, está dispuesto con el terminal de entrada I a un potencial positivo. Sin adentrarnos en la física de los semiconductores vamos a partir del supuesto de que, en la situación mostrada, el transistor se comporta como un interruptor cerrado y el terminal U está a tensión cero. El caso opuesto es el de la figura 1c, con el terminal I puesto a potencial 0, por lo que es como si el interruptor estuviera abierto, con el terminal U al potencial positivo +V. Así pues siempre estaremos manejando dos situaciones (estados) distintos: 0 por un lado y

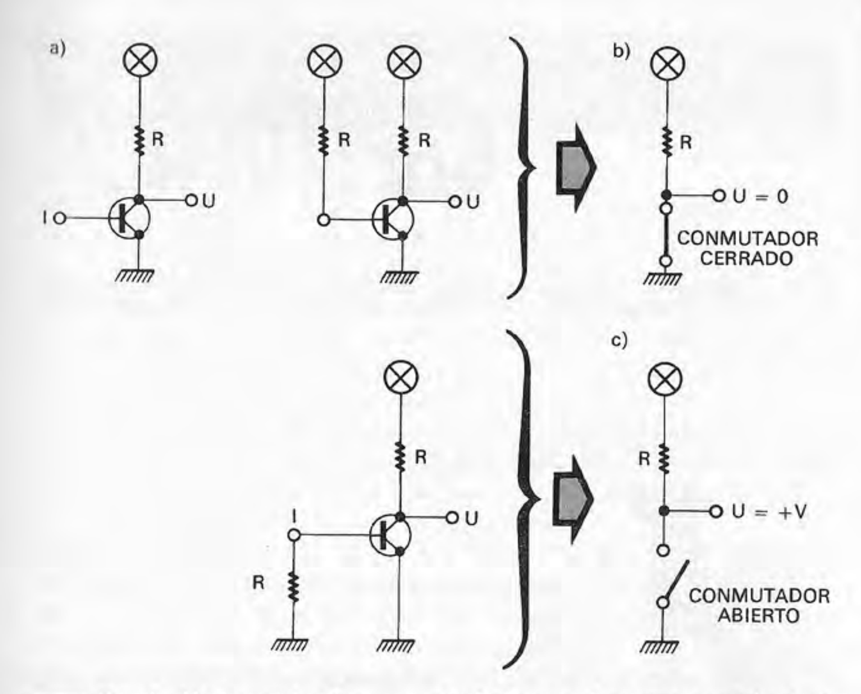



*Fig. 1 - Un transistor (como el de la figura a) es una especie de interruptor: si se aplica una señal positiva a su entrada de base, la salida pasará a 0 Voltios (b), mientras que si la entrada está conectada a masa, la salida estará a + V (c).*

+V por otro. Es por tanto un funcionamiento binario (de dos estados).

Estas breves consideraciones nos hacen reflexionar sobre el principio fundamental base del funcionamiento de cualquier ordenador, desde el más pequeño al más grande:

Cualquier acción realizada por los circuitos que constituyen la máquina, siempre será un tratamiento más o menos complejo de señales binarias. En pocas palabras, toda la electrónica digital está basada en conmutaciones de tensión del tipo "encendido/apagado" o "presencia/ausencia de tensión en un hilo". Todos los datos tratados en un ordenador son única y exclusivamente impulsos de tensión. Los denominados circuitos "lógicos", sólo pueden distinguir los estados de presencia o ausencia de tensión en sus entradas, y en función de dichos estados dar un valor binario a su salida.



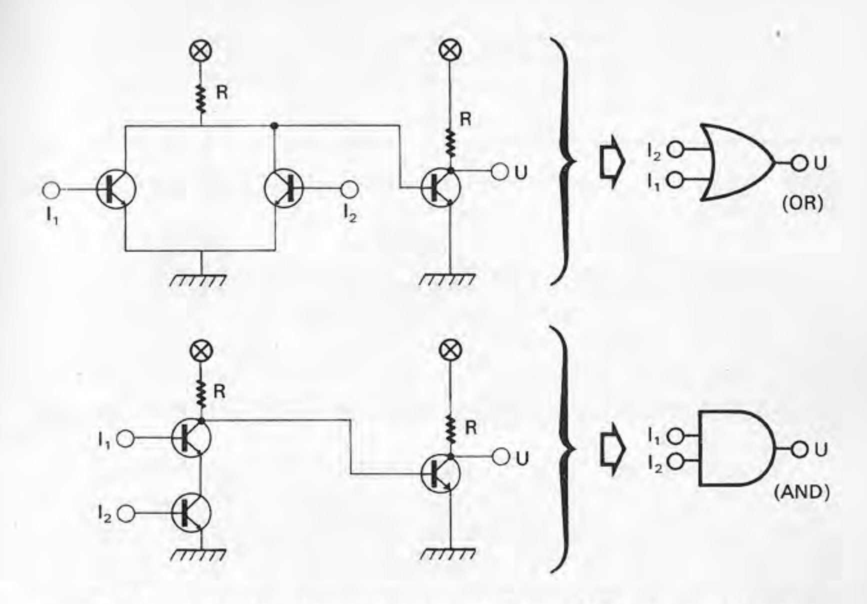



*Fig. 2 - Funciones lógicas simples (OR y AND) realizadas con transistores.*

Incluso el más complejo de los circuitos digitales se puede descomponer en bloques secundarios cada vez más sencillos, hasta su "disección" completa en funciones lógicas elementales. Estas últimas son las famosas NOT, AND y OR. Por lo que respecta al hardware los nombres de estas funciones van asociadas a las "puertas" (conjunto de transistores y resistencias) que las originan. Resulta lo mismo, por ejemplo, hablar de OR o de una puerta OR. Observe la figura 2. Con ella hemos pretendido satisfacer la curiosidad de aquellos lectores que se puedan preguntar cómo podría obtenerse, con simples transistores como el de la figura 1, un circuito lógico del tipo AND u OR. Si se fija verá también que en la figura 1 lo que se muestra es la realización de una puerta NOT. La figura 3 ilustra, con clásicos ejemplos de electricidad, el comportamiento de las funciones AND y OR: el circuito AND enciende la lámpara si, y solamente si, los dos interruptores están cerrados, es decir están a nivel de tensión positivo, lo que hablando en términos digitales equivale a decir que las dos entradas son "verdaderas", o también que están a "nivel lógico 1" (vea la tabla 1 para aclarar

| interruptor | tensión | valor lógico | nivel lógico |
|---|---|---|---|
| cerrado<br>abierto | positiva<br>negativa | verdadero<br>falso | alto (1)<br>bajo (0) |

*Tabla 1 - Equivalencias de las distintas terminologías en un sistema binario.*

estos términos). Basta que una sola de las dos entradas esté al nivel lógico "0" para tener a la salida un nivel lógico "0". Por el contrario, la función OR muestra que basta que un solo interruptor cerrado ("1") para tener la lámpara encendida; las dos entradas deben estar a "0" para tener también "0" a la salida.

El empleo de "1" y "0" para indicar que una entrada está a un potencial +V (+V es el valor genérico de la tensión de alimentación, que suele ser de +5V) o a potencial 0 es algo característico de la lógica digital, porque simplifica el tratamiento y es de comprensión inmediata. En lo sucesivo, seguiremos siempre dicha nomenclatura.

La figura 4 muestra las denominadas "tablas de verdad" de las funciones AND, OR y NOT. Estas tablas representan todas las combinaciones posibles de las entradas de la puerta, y el resultado correspondiente en la salida. Debajo de cada una de las tablas está dibujado el símbolo lógico de la puerta correspondiente; éste suele emplearse en todos los esquemas digitales, como se ve en la figura 5a que, a título de curiosidad, ilustra el esquema de algunas funciones

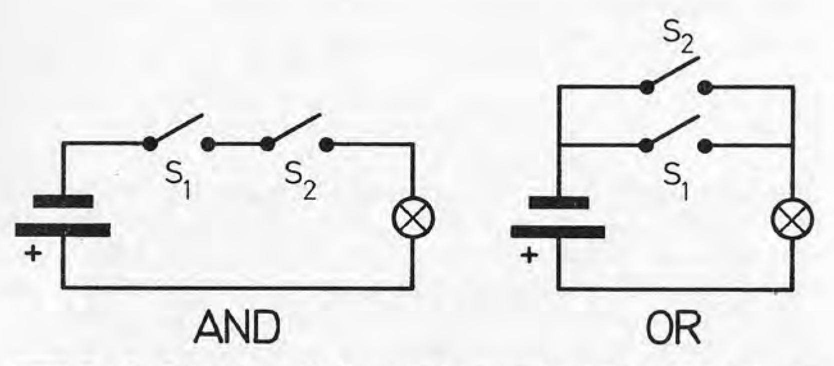



*Fig. 3 - Representación funcional con interruptores de las funciones lógicas AND y OR.*



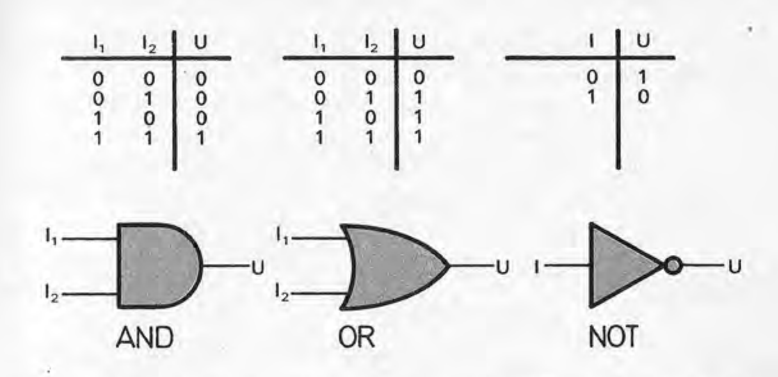



*Fig. 4 - Tablas de verdad de las funciones lógicas AND, OR y NOT.*

lógicas complejas que son realizables con las puertas que acabamos de describir. Estas puertas no se presentan solas, sino agrupadas varias en un solo chip, con características normalizadas; son los circuitos integrados TTL.

Otros ejemplos de bloques lógicos más complicados los puede encontrar, por ejemplo, en el manual técnico ("Hardware Manual") de su propio ordenador personal.

El recorrido a realizar para llegar al microprocesador a partir de las puertas lógicas OR, AND y NOT es más bien largo y no disponemos de bastante espacio para abordarlo. Por ello consideramos suficiente haber adquirido al menos estos conceptos básicos. Si desea profundizar más en ellos puede emplear los libros que le recomendamos en la bibliografía o bien dejarse guiar por las reseñas bibliográficas de las revistas especializadas (Elektor, Tu Micro...).

## LSI: La tecnología que ha permitido la informática personal

Si, como todos sabemos, un transistor es un "bicho" pequeño pero "visible" imagínese el espacio que ocuparían cincuenta mil (¡ha leído bien, cincuenta mil!) de ellos. Afortunadamente, la tecnología de los quince últimos años nos ha echado una mano y con la integración a cada vez más amplia escala, se ha hecho posible la realización de dispositivos muy complejos, manteniendo unas dimensiones muy reducidas.

a)  b)

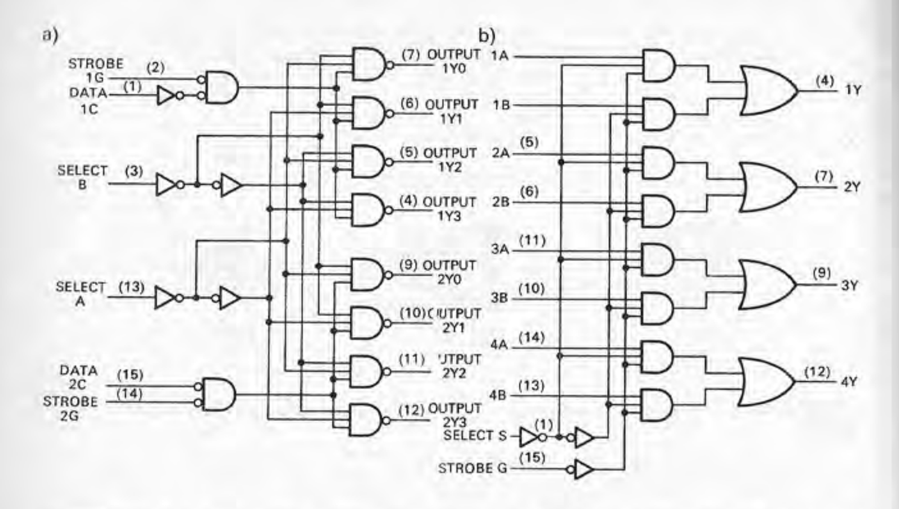



*Fig. 5 - Algunos ejemplos de circuitos digitales complejos, realizados con circuitos integrados TTL.*

¿Cómo es posible? La respuesta está ya ante nuestros ojos, en la tarjeta principal del ordenador personal que acabamos de abrir y que está llena de esos dispositivos cuyo color y forma les ha valido el mote de "cucarachas". Cada uno de estos circuitos, denominados "integrados" (C.I.), es una pastilla, bien de plástico o de material cerámico, que contiene una diminuta placa de silicio, material semimetálico (o semiconductor) con el que están construidos los transistores. La "gracia" de estos C.I. reside en que en lugar de utilizar un trozo de silicio para cada transistor y luego conectarlos todos juntos exteriormente, utilizan una placa fotograbada que contiene la disposición de todos los transistores, incluidas las conexiones correspondientes. Es algo así como el tallador que obtiene una mesa artística a partir de un solo trozo de madera, representando todas las figuras juntas, en lugar de realizar las escenas por separado, una a una y luego unirlas.

En el proceso de diseño de los circuitos integrados, unas máquinas especiales "traducen" los esquemas lógicos (realizados en papel con los elementos AND-OR-NOT antes considerados) a sus esquemas eléctricos (basados en transistores) y estos a trazados adecuados que se practicaran en el chip de silicio con un grabado fotolitográfico. La ventaja es enorme, puesto que el espacio que ocupan los elementos activos se reduce, por medios fotográficos, a dimensiones del orden de las micras; esto hace posible la integración de miles de funciones lógicas. Para las simples puertas que examinamos anteriormente, se usa la integración a baja escala (SSI = Small Scale Integration), puesto que no son muchos circuitos los que han de colocarse en el chip: en la tarjeta principal del ordenador personal las puertas lógicas suelen estar incluidas en forma de circuitos integrados de 14 patillas. Para las funciones complejas, características de los circuitos integrados algo más grandes se emplea una integración a media escala (MSI) que permite entre 10 y 1000 puertas por chip; los integrados más complejos, grandes y costosos, tales como las memorias, los controladores de E/S y, naturalmente, la misma CPU, se fabrican con técnicas de integración a gran escala o a muy gran escala (LSI y VLSI, respectivamente) que permiten hasta 10.000 puertas lógicas la 1.ª y más la 2.ª.

Un detalle que hay que tener presente es que las dimensiones de la cápsula no dependen tanto de la complejidad del chip como del número de funciones que precisen comunicarse con el exterior como entradas o salidas; es decir, desde el punto de vista práctico el tamaño depende del número de patillas (pines) necesario. Un microprocesador suele estar confinado en una cápsula de 40 patillas, porque es elevado el número de señales que debe controlar, tanto de entrada como de salida (dentro de poco veremos cómo lo hace). En la figura 6 se puede ver la fotografía del chip de silicio en el que está integrado el conocido microprocesador Z80. En realidad, la placa mide aproximadamente medio centímetro de lado.

## ¿Cómo funciona un microprocesador?

Ahora que estamos familiarizados con las funciones lógicas fundamentales y que nos hemos introducido en la tecnología de fabricación de los dispositivos digitales, podemos pensar en la construcción de nuestra CPU. El "truco" consiste en olvidarnos de las puertas AND, OR y NOT, ha-

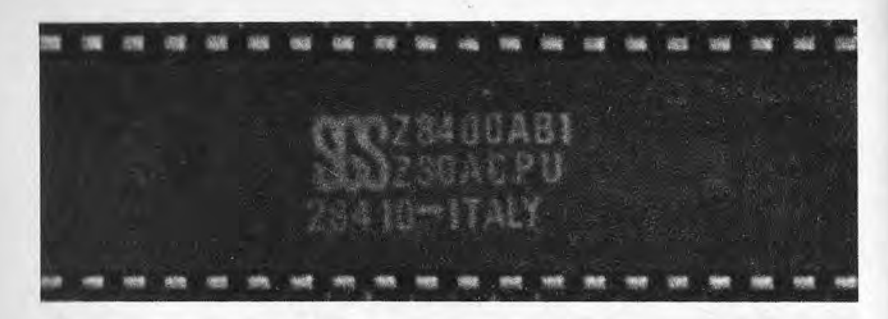



*Fig. 6 - Fotografía del chip de silicio que constituye el microprocesador Z80.*

ciendo el razonamiento siguiente: "utilizaremos solamente bloques lógicos cuyo nombre coincidirá con el de la función que realicen y formados por la unión de tantas puertas elementales como sean necesarias".

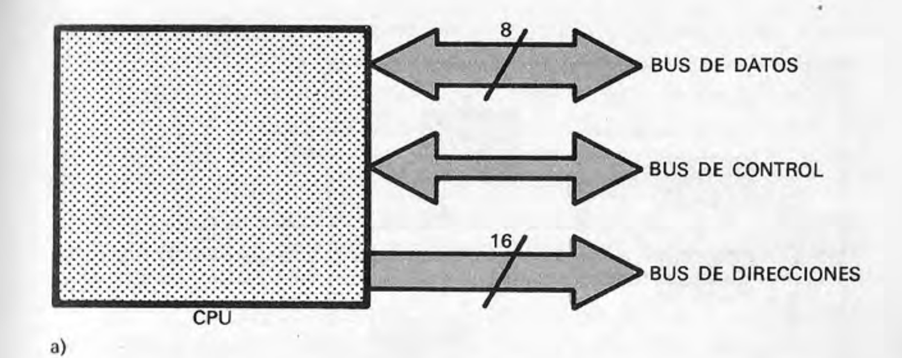
Quizá no sea un método habitual construirse una máquina para comprender cómo funciona pero, en este caso, veremos cómo el estar obligados a pensar en los dispositivos necesarios en una CPU, nos ayudará mucho a la comprensión del funcionamiento del conjunto.

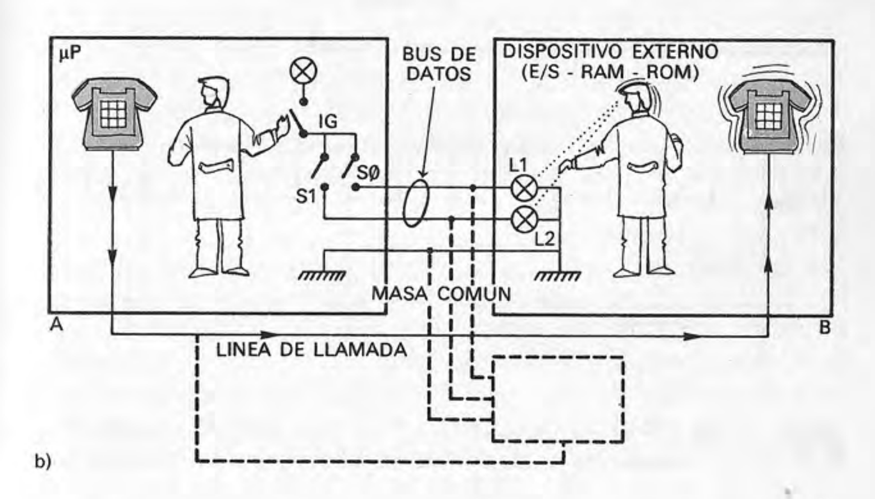
Con este supuesto previo, dibujamos lo que aparece en la figura 7a. El rectángulo define los límites del microprocesador: por ahora está vacío, pero lo iremos rellenando poco a poco.

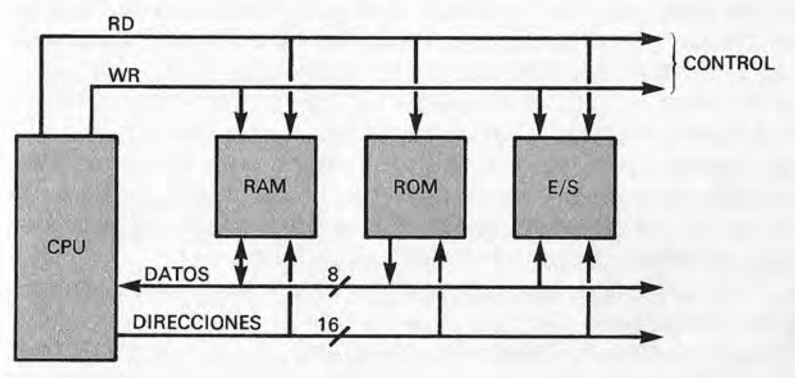
Comencemos por examinar las conexiones externas, que se traducirán en otros tantos terminales o patillas. Tomamos ocho hilos de conexión y los agrupamos juntos, creando el denominado "Bus de datos" (en inglés "Data Bus"). A través del Bus de datos podrán llegar a la CPU señales (datos) procedentes de los dispositivos que le conectaremos y, recíprocamente, la CPU podrá, a través de él emitir señales.

En la figura 7b se ilustra, esquemáticamente, la transmisión de un dato en el bus. Para una mayor sencillez, hemos considerado que el bus tiene solamente dos hilos, pero la explicación se puede extender a ocho hilos sin problema alguno. Pues bien, el microprocesador sería el hombre que está en el recinto A, y el dispositivo exterior conectado al microprocesador, que puede ser RAM, ROM o E/S, es el que está sentado en el recinto B. Cada línea puede dar tensión

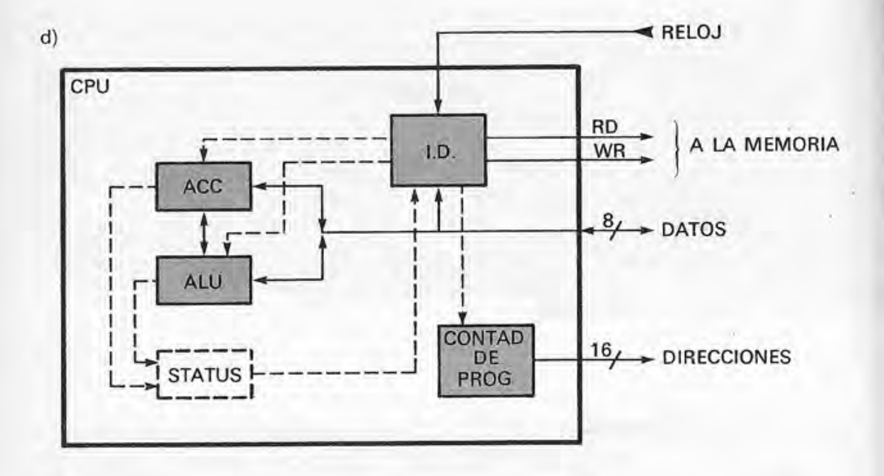




d)




e)

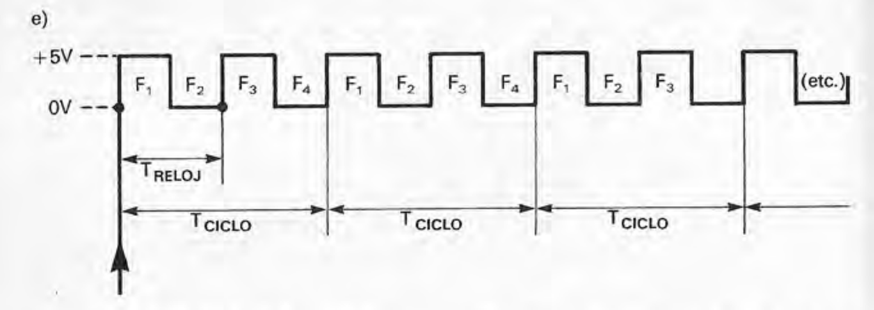



*Fig. 7 (a, b, c, d y e) - Construcción de una CPU a partir de las funciones necesarias y de los principales bloques internos (véase texto).*

a una lámpara del recinto B, cuyo encendido está controlado desde el recinto A, es decir, desde el microprocesador. Las referencias de masa son comunes, tal como sucede en la realidad. Si el microprocesador ($\mu$P) quiere enviar un dato al dispositivo, mediante una operación que se denomina "escritura del dato", el dispositivo deberá, ante todo, ser avisado a fin de que esté preparado para recibir el dato. Esta operación de aviso se realiza con una línea adicional, representada en nuestra ilustración por el teléfono.

El teléfono está sonando en B y éste se encuentra ya preparado para recibir el dato, operación que, en nuestro caso, consiste en observar el estado de las lámpara mientras la llamada siga produciéndose (es decir, hasta que deje



de sonar el teléfono). Ahora A deberá posicionar, a su elección, los interruptores de datos S0 y S1 y, luego, accionar el interruptor general IG para dar tensión simultáneamente a todos los interruptores. De forma inmediata, B "verá" una de las cuatro combinaciones posibles de "encendido/apagado" de las dos lámparas y tomará inmediatamente nota de dicho estado porque, transcurrido un corto período de tiempo, A cortará la llamada y dejará libre a B para dedicarse a otra actividad.

Naturalmente, nada impide al microprocesador llamar inmediatamente después a otros dispositivos conectados (indicados en la figura con líneas de trazos) o volver a llamar al propio B. Si ha prestado atención hasta aquí le habrá "chocado" que en el ejemplo de la figura 7b se muestre un bus de datos unidireccional. Por el contrario, en la realidad, el bus de datos es bidireccional; esto significa que también el dispositivo B tiene sus interruptores de preselección y A sus lámparas de visualización del dato. La única limitación a esta "bidireccionalidad" está en que, incluso para leer un dato, será siempre el microprocesador quién llame al dispositivo exterior, quien controle todas las transferencias y, en cualquier caso, quien tendrá bajo su control a todo el sistema electrónico. ¿Cómo sabe el dispositivo exterior, en el momento en que se le llama, si la CPU solicita una lectura o una escritura? Sencillamente: mediante dos líneas de control, denominadas RD y WR (que corresponden, respectivamente, a la petición o habilitación de lectura y escritura). La CPU, mientras avisa a un dispositivo, pone una de las dos líneas a "1"; el dispositivo sabrá entonces qué debe hacer: si proporcionar el dato que tiene a la CPU para que esta lo lea o admitir el presente en el bus de datos para una escritura. Las líneas de control están normalmente agrupadas, como los datos, en un bus llamado Bus de control.

Cuando hay numerosos dispositivos conectados, resultaría incómodo y poco práctico usar una línea individual en el microprocesador por cada dispositivo. Volviendo al ejemplo anterior, podemos hacer que cada dispositivo tenga su número de teléfono, distinto del de los demás. El número de teléfono se denomina "dirección"; esta es la razón por la que, ya en la figura 7a, habíamos dibujado otros 16 hilos a la salida del microprocesador: estos forman otro bus que se denomina "Bus de direcciones" (en inglés "Address Bus").

Cuando el microprocesador llama a un dispositivo, para escribir o leer un dato, emite el código de ese dispositivo, a través de las 16 líneas del bus de direcciones que están conectadas con todos los dispositivos exteriores. El código es leído por todo los dispositivos pero tan solo uno se "dará por aludido" al coincidir el valor transmitido con su "número".

Hagamos algunas cuentas: por una línea podemos transmitir un "0" (lámpara apagada) o un "1" (lámpara encendida); en dos líneas, las combinaciones son 00, 01, 10 y 11; en tres líneas, desde 000 à 111, tenemos ocho combinaciones, y así sucesivamente: el número de combinaciones posibles depende (en razón de la potencia de 2) de las líneas disponibles. Así, con 8 líneas tendremos $2^8 = 256$ combinaciones diversas y en 16 líneas podremos transmitir hasta $2^{16} = 65536$ códigos diferentes. Este será, pues, el número máximo de dispositivos que se puede conectar externamente a la CPU.

En realidad, por razones de economía y rapidez, muchos de los circuitos integrados del mercado (las memorias, por ejemplo) contienen en sí millares de dispositivos simples, haciendo más sencillo configurar luego el circuito.

El $\mu$P es todavía una caja vacía, pero ya sabemos que debe estar conectado al exterior mediante líneas dispuestas en grupos (buses).

El bus de datos permite el intercambio recíproco (bidireccional) de información, a través de 8 hilos, entre el microprocesador y los dispositivos exteriores. El bus de direcciones sale del microprocesador y es unidireccional; sirve para que el microprocesador emita el código del dispositivo con el que quiere dialogar, denominándose dicho código dirección. Finalmente, las líneas del bus de control sirven para que los dispositivos exteriores identifiquen el tipo de operación que la CPU quiere desarrollar.

Una observación: las transferencias de datos desde, y hacia la CPU y la transmisión de las direcciones desde la CPU, se realizan en paralelo; es decir, todas las líneas asumen simultáneamente la configuración preestablecida de "unos" y de "ceros" (¿recuerda el interruptor general del ejemplo anterior?), con lo que se evitan confusiones y retrasos.

En informática se dice que una información precisa (o "es" de) n bits cuando para transmitirla son necesarias n líneas. Así el bus de datos es de 8 bits y el de direcciones de 16. La palabra bit proviene de Binary digIT (dígito binario) y está universalmente aceptada. Igualmente lo está el término byte que se refiere a un conjunto de 8 bits (el bus de datos será de 1 byte y el de direcciones de dos).

La figura 7c representa la arquitectura genérica de un ordenador construido sobre la base de la CPU: los dos buses llegan a todos los dispositivos; el bus de datos es bidireccional, mientras que el de direcciones no lo es. Observe que también las líneas RD y WR están conectadas en paralelo a todos los dispositivos, aunque uno de ellos recibe solamente la línea de lectura RD (se trata de la memoria ROM y más adelante veremoa la razón de ello).

Con lo que sabemos podemos añadir algunas cosas al interior de la caja vacía. En la figura 7d se muestran cinco bloques, el "decodificador de instrucciones" (denominado ID en la figura), un contador de 16 bits que es el contador de programa o contador de pasos de programa, un dispositivo denominado "acumulador" y una unidad denominada "ALU", que es la abreviatura inglesa de "Unidad Aritmético-Lógica". El último bloque es un registro llamado de "estado" (status).

El ID (decodificador de instrucciones) es verdaderamente importante pues controla el funcionamiento interno del microprocesador. El bus de datos que está conectado al microprocesador llega también al ID, pero solamente en un sentido: el ID sólo puede leer el código de 8 bits que llega procedente del bus de datos no escribir algo en este. Vimos anteriormente que con ocho líneas se pueden representar 256 combinaciones de "encendido/apagado" o de "1/0": desde la 00000000 a la 11111111. Estos 256 códigos son utilizados para enviar órdenes "codificadas" desde el exterior a nuestro ID.

Esencialmente, el ID es un bloque lógico capaz de distinguir cada combinación individual de entre las 256 posibles y convertirla en una secuencia precisa de órdenes para los demás bloques existentes en el microprocesador. Cuanto más complejo, versátil y podemos decir que "virtuoso", sea el ID, tanto mejor funcionará el microprocesador. El fabricante de éste suministra normalmente un manual de programación en lenguaje máquina, que no es otra cosa que

una tabla que asocia cada uno de los códigos que podemos enviar al microprocesador con su efecto dentro de la CPU.

Ejemplo: el manual de la CPU 6502 indica que para el código 10101001, el ID obligará al acumulador a guardarse el dato que aparezca inmediatamente después en el bus de datos.

El contador de programa (PC = Program Counter) tiene la función de escribir en el bus de direcciones un código de 16 bits que abrirá el acceso a un dispositivo determinado. El PC no emite estos códigos de direcciones de forma aleatoria, sino que sigue las órdenes del ID, que le comunica en el momento oportuno cuál será la dirección de partida y cuándo y en qué magnitud deberá variar dicha dirección. Asimismo recibe también la orden de cuándo escribir su contenido en el bus de direcciones.

El acumulador es un bloque bastante simple; consiste en una memoria que lee los datos procedentes del bus y los almacena para que puedan utilizarse en operaciones posteriores. Este tipo de memorias simples que forman parte de la CPU se denominan "registros" de la CPU. En nuestro microprocesador-ejemplo sólo hay un registro, pero en la realidad son varios los registros que contiene la CPU, lo que aumenta su versatilidad y potencia.

Finalmente, sólo nos queda el bloque denominado ALU. En la ALU se realizan "cosas de locos" con los "unos" y "ceros" del dato que llega por el bus desde el exterior o bien desde el acumulador: los "unos" pueden transformarse "sádicamente" en "ceros" y viceversa; dos combinaciones de 8 bits pueden sumarse o restarse según las reglas del Algebra de Boole o de la Aritmética; un dato puede compararse con el contenido en el acumulador, etc.

Naturalmente, también en este caso todas las operaciones aritméticas (sumas, restas, etc.) o lógicas (comparaciones, conmutaciones 0/1, etc.) están estrechamente supervisadas por el ID que, dentro de la CPU, tiene un gran trabajo que hacer.

## *La señal de reloj (clock)*

Las tareas realizadas por todos los bloques internos de la CPU, bajo el control del ID, se sincronizan desde el exte-



rior con un ritmo impuesto por el denominado "reloj del sistema" (en inglés "clock"); se trata de una señal que se emplea como patrón de tiempo y sincroniza todas las operaciones realizadas por los diversos bloques integrados en la CPU. En la misma figura 7d, la señal de reloj llega al ID y éste actúa como si tuviese en la mano un cronómetro con el que controlara la sincronización de las fases de trabajo de cada bloque supervisado. Una señal de reloj clásica es una tensión que varía, de forma cíclica, entre 0 y +5V, con una forma de onda cuadrada, como en la figura 7e. Su frecuencia se calcula observando cuántas veces por segundo se repite la forma básica denominada "ciclo" y representada por "Tclock" en la figura 7e. Si el reloj tiene una frecuencia de 4 MHz, en cada segundo proporcionará cuatro millones de ciclos, por lo que un ciclo durará exactamente 1/4.000.000 = $250.10^{9}$ seg. = 250 ns (nanosegundos = milmillonésimas de segundo). Cuanto más elevada es la frecuencia de reloj, tanto más grande es la velocidad de ejecución, pues el ciclo durará menos.

## *Funcionamiento interno de la CPU durante un ciclo*

Ahora sabemos que el reloj suministra una temporización de referencia. Veamos cómo se desarrolla el trabajo durante un "ciclo de máquina" (no tiene porqué coincidir con un "ciclo de reloj"). Examinemos las figuras 7d y 7e: la primera para las conexiones entre los bloques de la CPU y hacia el exterior, y la segunda para el reloj. Observará que hemos considerado dos ciclos contiguos de reloj, llamando F1, F2, F3 y F4 a los cuatro semiciclos consecutivos. El ciclo de máquina es "Tcycle" y dura, en nuestro ejemplo, el doble del período de reloj, lo que significa 500 ns para un reloj de 4 MHz.

Fijamos un comienzo para la base de tiempos en donde suponemos que el microprocesador toma vida de golpe y empieza a trabajar. Hemos llamado "I" a este instante inicial, que coincide con el comienzo de la primera fase, F1. El ID establece inmediatamente el contador de programa con la dirección de partida, cuyo valor característico varía de una CPU a otra (en la 6502 sería la 65.529). En la fase F2, el ID carga el valor del contador de programa en el bus de di-

recciones y, al mismo tiempo, activa la línea RD para solicitar una lectura del dispositivo seleccionado. Este último, reconocida su dirección en el bus, reacciona inmediatamente y emplea el período F3 para encontrar el dato solicitado. En F4, el dato es situado por el dispositivo en el bus de datos y leído por el ID.

Este dato procedente del exterior es decodificado por el ID para saber qué operación debe ejecutar; en la fase F1 del ciclo de máquina siguiente comienza a controlar todas las operaciones internas relacionadas con esa ejecución en particular. Puede darse el caso de que basten dos ciclos de máquina para completar todas las operaciones; entonces, el tercer ciclo será en realidad el primero dedicado a la siguiente operación. Para muchas operaciones, sin embargo, no bastan dos ciclos de máquina, sino que pueden precisarse tres, cuatro e incluso más. En cualquier caso, siempre es el primer ciclo de máquina el dedicado a la adquisición del dato clave para el ID. Dicho dato toma el nombre de Código de Operación ("OP.CODE") y sirve para seleccionar la serie de operaciones internas necesarias. A continuación facilitamos una breve lista de algunas operaciones comunes a todas las CPU:

- Carga inmediata de un valor en acumulador (el valor a cargar forma parte de la propia operación).
- Desplazar un dato desde un dispositivo exterior a otro (desde una dirección a otra).
- Sumar un valor que llega desde el exterior a un segundo dato ya existente en el acumulador.
- Comparar un dato que llega desde el exterior con un dato ya existente en el acumulador.

Proporcionar una instrucción al microprocesador significa, ante todo, enviar el código de operación que la define. Para completar la instrucción debemos suministrar todos los datos a los que hace referencia enviándolos inmediatamente después del código de operación. Estos datos que completan la instrucción son los denominados OPERANDOS. En definitiva, una instrucción genérica de $\mu$P tiene el formato siguiente:

CODIGO DE OPERACION [OPERANDO] ... [OPERANDO]



donde el código de operación es la única parte siempre necesaria.

Habíamos dejado el ID tras el primer ciclo de máquina preparado para controlar la ejecución de la instrucción especificada por el código de operación enviado. En este punto, podemos seguir la ejecución de algunas de las sencillas instrucciones antes citadas.

- Primer caso: "carga inmediata en el acumulador del dato que sigue".

En el primer ciclo de máquina, ya examinado, el ID comprendió la instrucción; en el ciclo siguiente su tarea consistirá en: ordenar al contador de programa que emita una nueva dirección igual a la anterior +1 (durante F1); activar, en F2 la línea de control RD para una lectura del dispositivo que ha de proporcionar el dato (F2); en F3 seleccionar el dispositivo y depositar el dato en el bus y en F4 cargarlo en el interior del acumulador que, mientras tanto, fue avisado por el propio ID. Conclusión: la instrucción se completa en dos ciclos de máquina.

- Segundo caso: "Suma al contenido del acumulador el dato que sigue inmediatamente (el resultado queda en el acumulador)".

Después del primer ciclo de máquina, sigue un segundo ciclo idéntico al comentado por la instrucción antes considerada, con la salvedad de que el dato que llega no se carga en el acumulador, sino en la ALU y que en F4, el ID ordena a la ALU que sume su contenido con el del acumulador. Al final de F4 el resultado queda en el acumulador.

**Recapitulando:** La ejecución de cada instrucción está pues controlada por el bloque del "decodificador de instrucciones", que precisa un código clave para identificar cada instrucción.

Dicho código se llama "código de operación" y hemos visto que normalmente va seguido por otros datos, denominados operandos, que completan la instrucción.

Cuando se conecta la CPU, tendremos que proporcionarle inmediatamente un código de operación ejecutable. Finalmente, hemos visto como una instrucción se lleva a cabo en varios ciclos de máquina; una vez acabada su ejecución la CPU puede iniciar inmediatamente la siguiente.

## *Carácter secuencial de la ejecución de operaciones por la CPU*

Por cuanto hemos visto hasta ahora, parece claro que la CPU es capaz de ejecutar todas las instrucciones a condición de que sea respetada su naturaleza secuencial: la CPU ejecuta una instrucción utilizando el número requerido de ciclos de máquina, y es preciso que termine la ejecución de una instrucción antes de iniciar otra. Si suministramos una secuencia correcta de códigos de operación seguidos cada uno por su operando (u operandos) adecuado, la CPU conseguirá ejecutar de forma correcta el "programa" que le hemos proporcionado.

Nos encontramos en la situación de la figura 8 que, si bien es un poco "festiva", indica claramente la relación entre la CPU, los dispositivos externos y las instrucciones. El bus de datos es una cinta transportadora que lleva los datos de forma secuencial hacia la CPU. Los datos están simbólicamente representados por grupos de cubos, el primero de

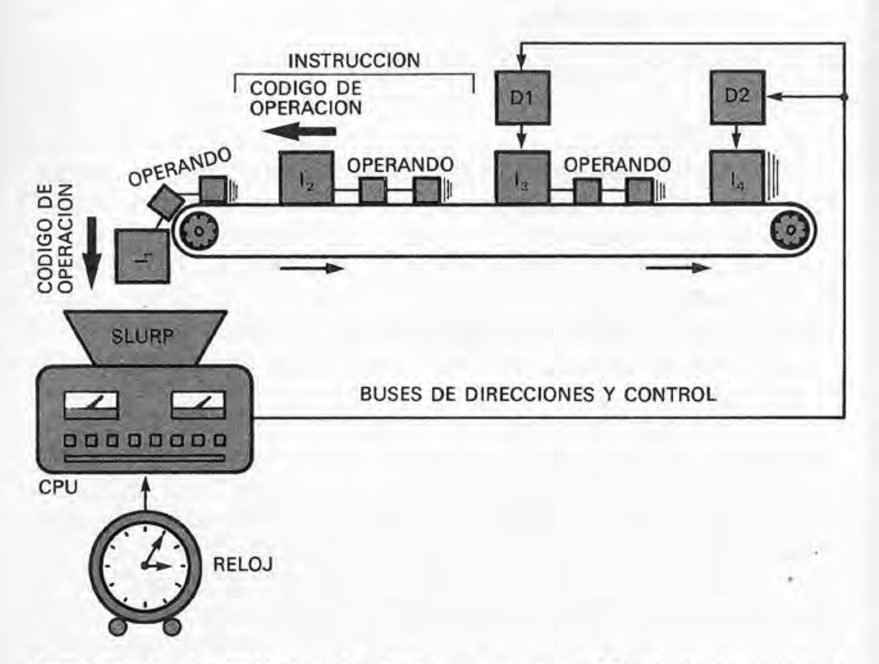



*Fig. 8 - La CPU absorbe de forma secuencial las instrucciones que le llegan procedentes de los dispositivos exteriores (memoria, E/S).*



los cuales es siempre el código de operación, mientras que los demás son (si existen) los operandos. Cada grupo es una instrucción, que se ve absorbida de forma secuencial por la CPU. ¿Quién introduce los datos en el correspondiente bus de datos?: los dispositivos externos, después de haber sido seleccionados mediante el Bus de direcciones y el de control. Todas las operaciones son "implacablemente" controladas por el reloj (vea el "despertador de la abuela"). Para no hacer demasiado complicado el dibujo, el bus de datos se ha representado como unidireccional.

Por lo dicho hasta ahora, podría parecer que la ejecución de toda instrucción trae consigo exclusivamente la lectura de datos en los dispositivos externos hacia la CPU, pero no es cierto. Podemos darnos cuenta, por ejemplo, siguiendo la ejecución de la secuencia "desplaza un dato desde el dispositivo 1 al dispositivo 2". Así veremos que es necesaria la bidireccionalidad del bus de datos.

En el primer ciclo de máquina, el decodificador de instrucciones (ID) reconoce el código de operación que corresponde a la instrucción "desplaza un dato de un dispositivo a otro". Para ejecutarla la CPU tendrá que conocer tanto el dispositivo en el que se ha de realizar la lectura como aquél en el que se va a escribir el dato. Sabemos que cada dispositivo tiene una dirección: pues bien, la CPU debe identificar la dirección de "lectura" y la de "escritura", que se denominan "fuente" y "destino". Estas dos direcciones deben especificarse como operandos de la misma instrucción de desplazamiento (movimiento). Nuestra instrucción deberá estar entonces constituida por:

- Un dato inicial de 8 bits (código de operación).
- Dos datos de 8 bits, esto es un dato completo de 16 bits, que defina la dirección fuente (primer operando).
- Dos datos de 8 bits, para la dirección de 16 bits de destino (segundo operando).

Es fácil comprender que en el segundo ciclo de máquina, la CPU adquiere la mitad del primer operando. En este punto, es bueno recordar que los dispositivos externos trabajan todos ellos con datos que tienen una longitud de solamente 8 bits, mientras que las direcciones son de 16 bits.

En el tercer ciclo, la CPU adquiere la segunda parte de

la primera dirección y el ID la capta y la coloca junto a la primera mitad.

En el cuarto ciclo, el ID situa esta dirección completa de 16 bits en el Bus de direcciones, llamando al dispositivo externo 1 y actuando el modo de lectura para obtener el dato. Al final del cuarto ciclo, el dispositivo fuente ha enviado el dato que es captado por la CPU y guardado en el acumulador.

Los ciclos quinto y sexto se dedican al ensamblado del segundo operando, por lo que al final del sexto ciclo el ID "tiene en el bolsillo" la nueva dirección de destino del dato. Finalmente, en el séptimo y último ciclo de máquina, el ID, con esta dirección, llama, en modo escritura, al dispositivo en el que se ha de guardar el dato. En este caso, el bus de datos está invertido y el flujo de datos se desplaza desde el acumulador de la CPU al dispositivo seleccionado para la escritura.

Conclusión: la instrucción de "movimiento" tiene una longitud de 5 bytes, es decir, utiliza 5 datos (incluido el código de operación) para quedar definida por completo, y su ejecución requiere 7 ciclos de máquina por parte de la CPU, de los cuales 6 son de lectura y 1, el último, es de escritura.

## *Registro de estado y decisiones de la CPU*

En la CPU de la figura 7d está indicado, en líneas de trazos, un registro muy importante: el registro de estado de la CPU (en inglés Status Register o Flag Register), del que hemos prescindido hasta aquí para que la exposición no se complicara demasiado.

Dicho registro pone de manifiesto, en cada momento, cuál es la situación dentro de la CPU. Sabemos que todas las instrucciones tratan los datos bajo la forma de "ceros" y "unos"; pues bien, puede haber alguna combinación que nos interese recordar con el fin de ejecutar de forma correcta la siguiente instrucción. Por ejemplo, el registro de estado recuerda si el último resultado que obtuvimos era un dato con todos los bits "ceros", si tenía el bit 7 puesto a "1" (número negativo), si existe un acarreo procedente de la suma que se acaba de ejecutar, etc. Cada una de estas situaciones, si se produce, "enciende" un bit del registro de estado, íntimamente conectado con el ID. De esta forma es posible que el ID pueda "tomar decisiones" sobre cómo ejecutar correctamente la siguiente instrucción basándose en los resultados de la instrucción anterior, reflejados en el registro de estado. La facultad de tomar decisiones es una de las características más importantes de un microprocesador y permite la escritura de programas complejos y versátiles, con instrucciones del tipo: "si el resultado de la instrucción es cero, haga esto; de no ser así, haga esto otro..."

## *¡Esto sí que es un programa!*

A los "supervivientes" del maratón precedente, les complacerá saber que la secuencia de instrucciones que hasta ahora hemos aplicado "brutalmente" a la entrada de la CPU, son realmente parte de un programa. Esta es pues, también la definición de "programa": secuencia de instrucciones ejecutables por la CPU. Ciertamente hay una gran diferencia entre un programa entendido como una secuencia de códigos máquina (OP.CODE+operandos) y un programa escrito "de forma inteligible", por ejemplo, en BASIC. En el segundo caso también se trata de una secuencia de instrucciones ejecutadas por el ordenador personal, pero hay una serie de pasos intermedios que facilitan la comprensión humana. Por ahora, nos contentamos con saber que la CPU sólo funciona perfectamente si le proporcionamos instrucciones "sensatas" una tras otra. Es evidente que, en la realidad no nos encargaremos de trasmitir personalmente a la CPU las instrucciones una a una y que, por supuesto, no lograremos seguir el ritmo del reloj. Por consiguiente, las instrucciones procederán de alguna forma de los dispositivos externos conectados a la CPU.

Intencionadamente no hemos profundizado hasta ahora en estos misteriosos "dispositivos", porque no creemos conveniente poner "demasiada carne en el asador" en un capítulo de por sí bastante árido para el principiante. Ha llegado el momento de revelar su identidad: dichos dispositivos son la memoria (RAM o ROM) y los circuitos de interconexión ("interface") con el exterior, es decir, los circuitos de entrada/salida (E/S, I/O en inglés). Sin estos circuitos integrados,

la CPU no podrá funcionar. En el siguiente capítulo se estudiarán con más detalle estos importantes dispositivos.

Antes de acabar el capítulo una observación: la exposición realizada sobre el funcionamiento de una CPU está basada en consideraciones generales y justificadas, aunque para los "Sherlock Holmes" de esta materia, aclaramos que las referencias tomadas un poco de acá y un poco de allá, se concentran en los microprocesadores Z80 (Zilog) y 6502 (Rockwell).

## *GLOSARIO*

SILICIO: Elemento de los más comunes en la tierra. Su símbolo químico es Si y se emplea como material base para la construcción de transistores y circuitos integrados. El silicio para tales usos, se presenta en cristales cilíndricos homogéneos y purísimos, que se cortan en rebanadas (como un embutido). Cada una de estas rodajas (u obleas), toma el nombre inglés de "wafer". En estos soportes se imprimen por medios microlitográficos y de fotograbación el interior de los chips; finalmente, se separan cortando la pastilla soporte y realizando su montaje en las cápsulas, para llegar al chip definitivo.

VLSI: Abreviatura de la expresión inglesa "Very Large Scale Integration" (Integración a muy gran escala). Es el nivel de complejidad de circuitos integrados tales como microprocesadores, memorias e interfaces de E/S. (La integración a pequeña, mediana y gran escala tienen las abreviaturas SSI, MSI y LSI).

CHIP: pastilla de silicio en la que están fotograbados los circuitos lógicos, preparados para uso inmediato, y alojada en cápsulas integradas.

ALGEBRA DE BOOLE: Serie de reglas que rigen las relaciones y operaciones entre valores lógicos; es decir, entre hechos que se pueden representar por "ceros" y "unos". Las operaciones lógicas fundamentales son: AND, OR y NOT. Con estos circuitos lógicos básicos se hacen circuitos más complejos, tales como la ALU (Unidad Aritmético-Lógica).

CICLO DE MAQUINA: Es el mínimo período de tiempo empleado por la CPU para adquirir un dato de la memoria o escribirlo en la misma. Se mide en número de ciclos de reloj.

CICLO DE RELOJ: Período de la onda cuadrada procedente de un dispositivo exterior, que sirve para sincronizar las funciones del μP.

BIT: Unidad de información. Puede tomar los valores "0" y "1".

BYTE: Agrupación de 8 bits. Los microprocesadores más conocidos son de 8 bits porque tienen el bus de datos de ese tamaño; otras unidades de CPU más modernas son de 16 bits al ser su bus de datos de 16 líneas (16 bits).

WORD (palabra): Conjunto de bits que pueden ser manipulados por el μP en una sola operación. Según la CPU la palabra puede ser de 1, 2 ó 4 bytes.

Z80: Conocida CPU producida por Zilog. Se utiliza con gran frecuencia en los ordenadores personales de gestión que adoptan el sistema operativo CP/M.

6502: Clásica CPU adoptada por Apple, que la ha empleado en todos sus ordenadores personales (con la excepción del LISA y el Macintosh, en los cuales utiliza la CPU 68000). La fabrican Rockwell, Synertek y MOS. También es el "corazón" de ordenadores personales tales como Atari, Commodore VIC20 y C64.

8086/8088: CPU de 16 bits de Intel adoptada por IBM para su PC y por otros fabricantes de microsistemas de 16 bits. El microprocesador 8088 (usado en el IBM-PC) tiene el bus de datos exterior de 8 bits, mientras que el ordenador M24 de Olivetti emplea la más potente CPU 8086, compatible (en software) con el 8088.

68000: CPU muy potente, producida por Motorola, que se utiliza en los ordenadores personales más modernos. Tiene el bus de datos exterior de 16 bits, pero internamente los registros son de 32 bits, lo que permite un tratamiento muy rápido de los valores almacenados en memoria.

# CAPITULO III

## CIRCUITOS DE APOYO A LA CPU

En el capítulo anterior hemos tratado a marchas forzadas de asimilar como trabaja una CPU. No obstante, hemos podido darnos cuenta también de que una CPU no funciona por sí sola; de hecho, cada una de sus acciones está en estrecha correlación con la existencia de circuitos exteriores de apoyo, los denominados "chips periféricos". Estos tienen muchas funciones, pero las principales son proporcionar datos a la CPU, y aceptarlos de ella.

En el capítulo anterior, también clasificamos en tres categorías este tipo de chips: RAM, ROM y dispositivos de E/S. Veamos ahora con detalle sus funciones.

Comenzaremos con las memorias RAM, que pueden ser leídas y escritas por la CPU, prescindiendo de su función, todo cuanto digamos sobre sus conexiones y arquitectura tendrá validez también para la memoria ROM y para los dispositivos de E/S.

### RAM: "Random Access Memory". Memoria de lectura y escritura

En un sistema de microprocesador, la memoria tiene la función de recordar datos bajo la forma de bloques contiguos de bytes. Con frecuencia se habla en términos de bytes, puesto que, como explicamos con anterioridad, las

CPU de 8 bits tienen un bus de datos precisamente (¡casualidad, oiga!) de 8 bits y, por consiguiente, los chips periféricos conectados a ellas podrán suministrar, o admitir, datos completos de esta longitud.

Antes de examinar las conexiones del hardware propiamente dichas, nos concentraremos en dar una idea clara de la función que desempeña la memoria. Podemos compararla con un armario, provisto de una puerta principal con cerradura: si se abre, deja al descubierto una configuración interna constituida por varias filas de cajones, cada uno de los cuales tiene su propia etiqueta. Para llegar a un cajón, primero es preciso conocer el armario al que pertenece y seleccionarlo; esto es, hay que poseer la llave correspondiente a la puerta del armario. Además, debemos saber cuál es el cajón, dentro de ese armario, que queremos abrir para obtener el dato que contiene. Una vez abierto el cajón vemos su interior para leer el dato; entonces volveremos a cerrarlo quedando su contenido tal y como estaba. Evidentemente esta operación es una "lectura".

Si, por el contrario, una vez abierto el cajetín, vaciamos su contenido en un cesto y luego introducimos un nuevo dato, habremos efectuado una operación de "escritura": el nuevo dato ha sustituido al anterior. El proceso puede seguirlo en la figura 1.

De lo anteriormente expuesto se deduce que, una vez abierto el armario, cada cajón puede abrirse sin seguir ninguna regla ni respetar ninguna prioridad; esta es la razón

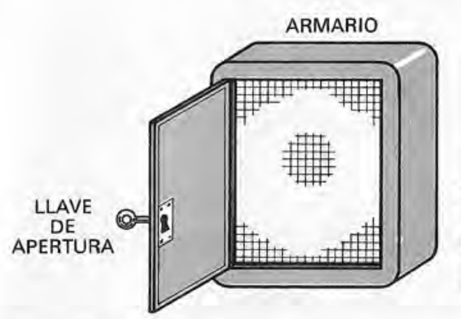



*Fig. 1 - Una memoria puede ser comparada con un armario que contuviera muchos cajones, cada uno de los cuales es accesible de manera independiente.*



por la que este dispositivo recibió la denominación de memoria de acceso aleatorio ("random access memory").

Si en este preciso momento, mediante una operación mental, cambiamos el mundo mecánico por el del hardware microelectrónico, pasaremos del armario al chip de memoria (por ejemplo, una RAM de 2048 bytes); a cada cajón corresponderá una celdilla individual de memoria de entre las 2048 disponibles. La memoria será impenetrable hasta que no poseamos la llave de acceso; en el armario abrimos la puerta y en el chip enviamos la señal de "selección de chip" (CS = chip select). Ahora que estamos en ello es conveniente aclarar algo sobre la terminología informática: una memoria de 2048 bytes se dice que tiene 2 kbytes. Aquí la K no indica ×1000 como en la física (kg = 1000 gramos) sino ¡mucha atención! ×1.024. La razón es que es el valor de la potencia de 2 más cercana a 1.000 ($2^{10}$ = 1024) y no olvidemos que los valores que podemos obtener en aritmética binaria son, forzosamente, suma de potencias de 2.

¿Quién genera la señal "chip select" (en lo sucesivo, denominada CS)? Naturalmente la CPU del sistema, aunque la operación no se produce de forma directa, sino a través de un hardware exterior que toma el nombre de "decodificador". Si el microordenador es la oficina, la CPU es el Jefe, las transferencias de los datos son realizadas físicamente por ...el inefable Sr. Ruíz (de forma muy rápida, por supuesto) y la memoria es el archivador, una operación de lectura puede tener lugar tal como se indica a continuación:

Jefe: "Ruíz, traígame una copia del capítulo 23 del expediente Ramírez (dato a buscar)".

Ruíz: Toma el libro mayor (decodificador) y encuentra: "expediente Ramírez: armario 3". Coge inmediatamente la llave correspondiente.

Ruíz: Abre el armario (Chip Select) y accede al cajetín 23, toma el expediente, lo fotocopia y vuelve a colocar el original donde estaba (operación de lectura completada).

Ruíz: Cierra el armario (cancela la selección de la memoria) y con las copias en la mano se dirige al despacho del Jefe (datos que llegan finalmente a la CPU).

Naturalmente, como suele ocurrir fuera del mundo del hardware, no hay ningún signo de agradecimiento por parte de la CPU.

La operación de escritura se produce de forma análoga, con la salvedad de que la orden de la CPU es: "ponga este expediente en el archivador". Ojo con esto, pues la operación de escritura tiene carácter destructivo, es decir, el contenido anterior de la memoria se perderá para siempre.

Pasando esta analogía al proceso real la cuestión sería: la CPU coloca en el Bus de direcciones la correspondiente a la posición de memoria que quiere leer (o escribir) y activa la señal RD (o WR) del Bus de control; el decodificador lee este valor y activa (CS) tan solo el chip de RAM que contiene dicha dirección. Entonces, la celdilla correspondiente copia sus datos en el Bus de datos si está activa RD o bien almacena los datos de dicho Bus en ella (si esta activa WR).

En este momento estamos en condiciones de analizar la figura 2. En ella la parte a) muestra la conexión de una típica memoria RAM de 2K × 8 y la parte b) indica el "patillaje" de circuitos integrados comerciales (recordemos que K es una constante que equivale a 1024). Se observa inmediatamente que hay 8 líneas para la transmisión del dato desde, y hacia, la memoria; de hecho, cada memoria RAM está conectada "en paralelo" al bus de datos. También está en paralelo la conexión a parte del bus de direcciones; la memoria sólo está conectada al número de líneas de dirección necesarias para generar todas las direcciones internas de sus celdillas, lo cual es posible gracias a la existencia de una posterior decodificación que se produce en el interior de la propia memoria.

Para distinguir físicamente 2048 celdas se precisan 11 líneas ($2^{11} = 2048$), desde A0 a A10. Sin embargo el bus de direcciones es de 16 líneas (desde A0 a A15) por tanto quedan 5 sin unir a la RAM. Hay que señalar que a la memoria RAM están conectadas las líneas "menos significativas" (de menor valor), mientras que las líneas "más significativas", desde A11 a A15, están conectadas al decodificador del sistema. Puesto que tenemos cinco líneas que controlan dicha decodificación, las combinaciones a la entrada podrán ser como máximo $2^5 = 32$; por consiguiente, el decodificador tendrá un máximo de 32 salidas de CS y podremos elegir una de entre esas 32 salidas independientes para utilizarla como selección de nuestro chip de memoria RAM. Así, por ejemplo, podemos hacer que la memoria RAM "responda" solamente cuando en el bus de direcciones la CPU sitúe una



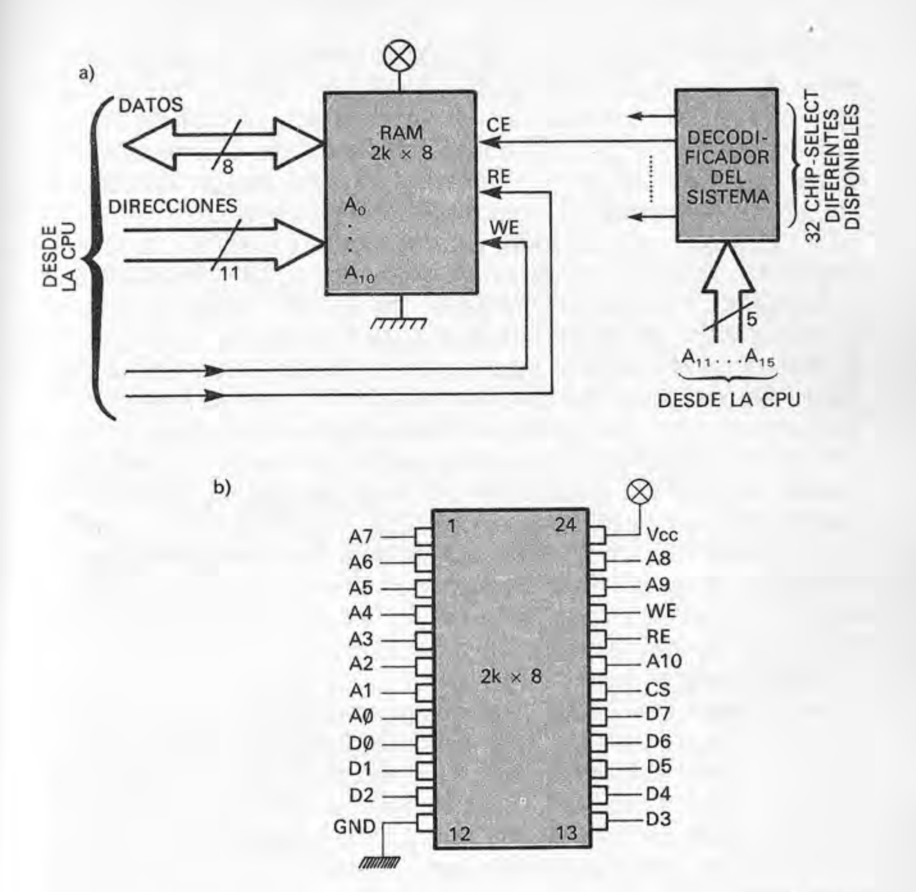



*Fig. 2 - Líneas de control, datos y direcciones de un chip de RAM (2K × 8) (a) y patillaje de un componente comercial (b).*

combinación de "ceros" y "unos" comprendida entre "0000 0000 0000 0000" ($0000 en hexadecimal —ver el glosario—) y "0000 0111 1111 1111" ($07 FF). Puede observar que, de forma intencionada, los cinco bits más significativos (más a la izquierda) de la dirección permanecen siempre fijos a "0000 0". Con cualquiera de estas disposiciones válidas el decodificador del sistema mantendrá activa una línea de salida que llamaremos "chip select 0" (CS0 que corresponderá a las líneas A11-A15 a 0). Para cualquier dirección comprendida entre estas 2048 (desde $0000 a $07FF) la memoria RAM será habilitada a través de la línea CS0. Como es evidente,

si CS0 direcciona 2Kbytes lo mismo ocurrirá con las otras líneas de CS, luego podremos abarcar 32 × 2048 = 65536 bytes, que es toda la memoria direccionable con un bus de direcciones de 16 bits ($2^{16}$ = 65.536). Conclusión: la CPU podrá dialogar con un máximo de 32 chips si cada uno es de 2K × 8 bytes. Evidentemente, a nadie se le impide utilizar chips de otra capacidad y poner en práctica una decodificación más compleja. Por otra parte, se pueden encontrar chips conectados a la CPU con capacidades de 4K, 8K y 16K por lo alto y, por lo bajo, de hasta algunos bytes, pues hay circuitos de E/S —como la VIA, PIA, etc.— que contienen muy pocas celdillas de memoria (normalmente 4,8 ó 16). (Seguramente habrá observado que, a veces "tachamos" el cero. Esto es una práctica normal en informática para diferenciarlo perfectamente de la O. Nosotros sólo lo hacemos cuando, por el contexto, se pueda prestar a confusión).

Volviendo a la figura 2, además de las líneas de alimentación (+ y masa) observamos dos líneas muy importantes en el control del chip: la línea de habilitación para lectura (RD) y la de habilitación para escritura (WR), ambas ya conocidas para el lector. Recordemos sólo que la CPU es el componente que sabe cuándo la operación a ejecutar es una lectura o una escritura, por consiguiente, será siempre la CPU quien active la línea de control necesaria. RD y WR (a veces llamadas RE y WE por Enable = habilitar, o sólo R y W) pueden ser ambas inactivas o bien una activa, y la otra inactiva, pero nunca pueden ser ambas activas a la vez. Para evitar que esto pueda ocurrir es muy normal introducir ambas en la misma línea de control (llamada entonces $R/\overline{W}$ o $\overline{R}/W$) de forma que cada una se considere activa con un valor de la línea (la que lleve el guión encima será activa cuando tengamos un 0).

Las figuras 3 y 4 muestran los diagramas de tiempos, o cronogramas, del desarrollo de un ciclo de lectura y de un ciclo de escritura entre la CPU y la RAM. Centrémonos en el primero (fig. 3), correspondiente a la lectura. La primera señal que se observa es la de reloj, controlando un ciclo cualquiera de máquina. La CPU establece el valor correspondiente en el bus de direcciones a la vez que activa la señal RE para indicar que va a realizar una lectura. El decodificador lee el valor del Bus de direcciones y, transcurrido un pequeño retardo Td, responde activando la línea CS co-



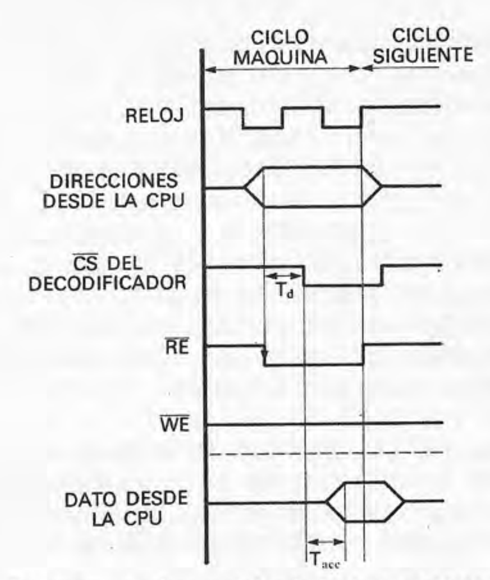


*Fig. 3 - Cronograma de un ciclo de lectura genérico.*

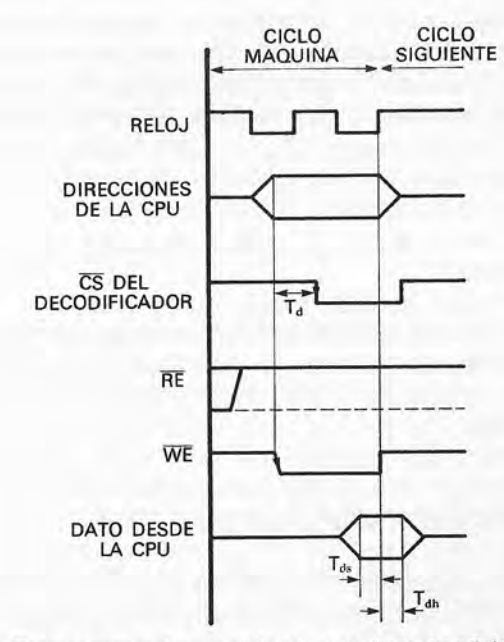


*Fig. 4 - Cronograma de un ciclo de escritura genérico.*

rrespondiente que entonces, habilita la memoria direccionada; esta sabe que la operación es de lectura al observar activa RE. La memoria RAM descifra el valor de las líneas del bus de direcciones aplicadas a sus entradas y, transcurrido un tiempo de acceso Tacc (varía según la tecnología de construcción, pero suele estar entre 150 y 400 ns para las memorias RAM más comunes) situa el dato localizado en esa celdilla en el bus de datos. Al finalizar el ciclo de máquina, la CPU puede "engullirlo" sin problema alguno. Inmediatamente borrará el bus de direcciones y la señal RE; el decodificador deshabilitará la señal CS, desactivando la memoria y quedando dispuesta para reactivarse durante el ciclo siguiente.

Si se trata de una escritura, todo se desarrollará como se indica en el cronograma de la figura 4. La CPU emite la dirección y el decodificador genera, después de un tiempo de retardo Td, la señal de Chip Select. Al mismo tiempo que la dirección, la señal WE se hace activa, por lo que la memoria RAM sabrá que "está por llegar" un dato procedente de la CPU. Esta última deberá mantener un cierto tiempo el dato en el bus de forma estable una vez identificada la celdilla que ha de modificarse; dicho tiempo se denomina Tds (Data-Setup). Eliminada entonces la señal WE el dato se graba en la RAM permaneciendo todavía en el bus durante un breve período de tiempo (Tdh = Data Hold —mantenimiento del dato—). Una vez acabado el ciclo, la CPU quita la dirección del bus correspondiente, desapareciendo CS; la memoria RAM se desactivará y la celda direccionada tendrá ya un dato nuevo.

Si ha seguido la explicación fijándose en las figuras, seguramente le habrá sorprendido que en varias señales cuando nosotros decíamos que se activaban en el cronograma pasaban del valor 1 al 0. Veamos cuáles son las razones de esta conducta.

## Señales activas: ¿a nivel alto(1) o a nivel bajo(0)?

Hasta ahora hemos tratado las señales de control (por ejemplo CS, WE, RE, etc.), con su nombre genérico, sin especificar el hecho de que la señal sea activa cuando esté a "1", o bien a "0". Aun cuando aquí seguiremos con esta nomenclatura (los gráficos y las ilustraciones servirán para disipar las dudas) creemos conveniente comentar lo que suele emplearse en catálogos y manuales técnicos: cuando una señal es activa en el nivel 1 se emplea su nombre sin más, en tanto cuando es activa en el nivel 0 se situa encima de su nombre un guión (p.e. $\overline{CS}$).

El atento lector habrá observado que las señales de control más comunes, tales como WE, RE, etc. están "activas" cuando su nivel lógico es "0", e inactivas si su nivel lógico es "1". La razón estriba en el uso de tecnologías (TTL, N-MOS) en las cuales los chips consumen menos energía si las entradas están al nivel lógico "1". Puesto que las señales que se utilizan en un microordenador son activas durante un tiempo muy pequeño frente al total, resulta conveniente utilizar señales activas con el nivel lógico "0" e inactivas con el nivel lógico "1", ya que el consumo será así inferior. Sin embargo, para los fines que perseguimos esta cuestión resulta irrelevante y podría complicar la explicación; por otra parte, en el capítulo anterior decidimos tratar los diversos componentes del hardware como "cajas negras", ignorando por completo la estructura interna. Así, seguiremos hablando de las diversas señales sin especificar constantemente si son activas con el nivel lógico "0" o "1"; simplemente lo explicitaremos cuando sea oportuno, sin más complicaciones.

## Algo más sobre la decodificación

Para no complicar en su momento la exposición, pasamos por alto la explicación del funcionamiento de un decodificador. Atemos ahora ese cabo suelto.

En la figura 5 se muestra un decodificador muy sencillo, constituido por 4 puertas AND y 8 puertas inversoras NOT (por sencillez, en el esquema las negaciones o inversiones están indicadas, según el convenio universal, por pequeños círculos dibujados en las entradas o en las salidas de las puertas). El decodificador queda definido físicamente por todo lo que está dentro del rectángulo de líneas a trazos y estará integrado en un solo chip. Este circuito distingue las cuatro combinaciones posibles en sus dos entradas,

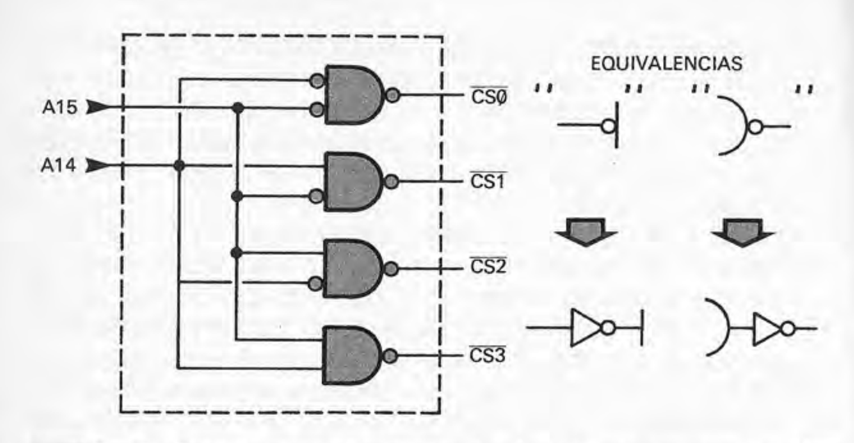



*Fig. 5 - Sencillo decodificador de 2 a 4, realizado con puertas AND e inversores.*

activando para cada combinación la línea de salida correspondiente.

En el capítulo segundo hemos descrito el funcionamiento de una puerta lógica AND y, por consiguiente, sabemos que su salida está al nivel lógico "1" solamente si todas las entradas están a dicho nivel. Resulta fácil comprobar, por ejemplo, que la salida CS0 es activada solamente si las dos entradas del decodificador, A14 y A15, están al nivel lógico "0", ya que luego serán objeto de negación y, por consiguiente, quedan ambas al nivel lógico "1", tal como se requería. La salida está, pués, a "1", pero pasará a "0" (observe el pequeño círculo de negación que transforma la puerta AND en una puerta NAND, esto es AND-NOT) y así obtendremos nuestra deseada señal de selección.

¿Por qué se utilizan precisamente A14 y A15 como entradas al decodificador? La respuesta es sencilla: A14 y A15 son las dos líneas de dirección más significativas del bus de direcciones; así, al utilizarlas, podemos subdividir físicamente, gracias a nuestro decodificador, toda la memoria direccionable en cuatro cómodos "bancos" de 16K × 8 bits (16Kbytes) cada uno.

Con el empleo de 4 chips de 16K bytes habremos implantado con facilidad toda la memoria (RAM y ROM) en nuestro ordenador, cuyo diagrama de bloques coincidirá con el de la figura 6. En esta figura, los cuatro chips están



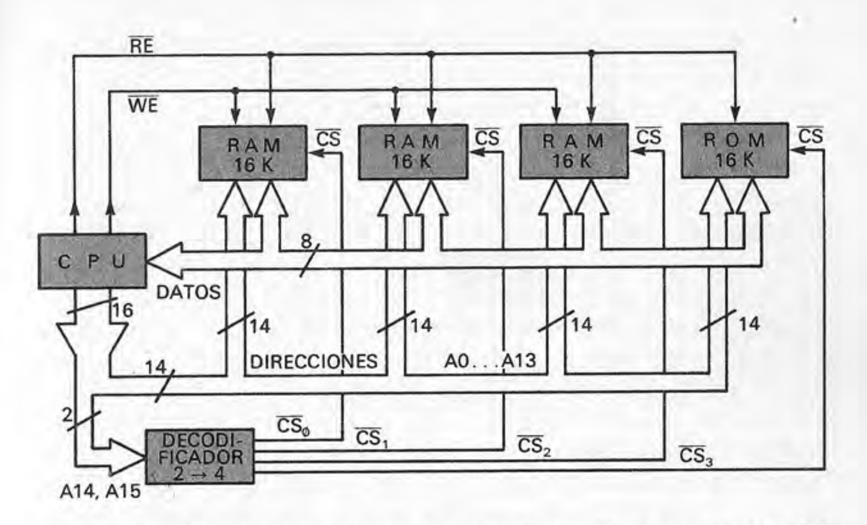
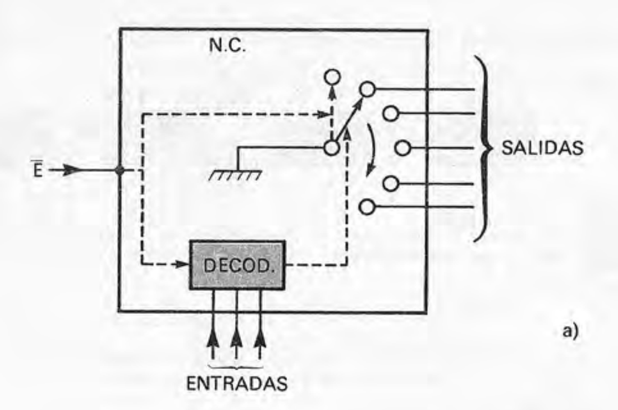



*Fig. 6 - Esquema completo de un diseño hardware con CPU, RAM, ROM y decodificador correspondiente para seleccionar, de forma correcta, 4 chips de 16 K cada uno.*

conectados en paralelo al bus de datos de 8 bits y al bus de direcciones (con la excepción de las dos líneas más significativas); 14 líneas bastan para identificar la celda objeto de llamada entre las 16.384 disponibles en cada chip ($2^{14}$). La selección es confiada a las señales de CS que llegan procedentes del decodificador; este último, a su vez, está conectado a las dos líneas sin usar del bus de direcciones (A14 y A15).

La figura 7a ilustra una representación mecánica de un decodificador: la configuración presente en la entrada posiciona el contacto del selector, activando la única salida válida. Es importante destacar la existencia de una entrada adicional que permite llevar el conmutador a una posición "neutra", en la cual todas las salidas son inactivas (nivel lógico "1"); dicha entrada se denomina "habilitación del decodificador" y suele estar activa para un nivel lógico "0". Con la no habilitación del decodificador, la situación de las entradas ya no tiene ninguna influencia sobre las salidas, que permanecerán según se dijo, todas ellas a nivel lógico "1".

Podemos crear muchos tipos de decodificadores, pero los modelos clásicos de la tecnología TTL son los de 2 a 4,



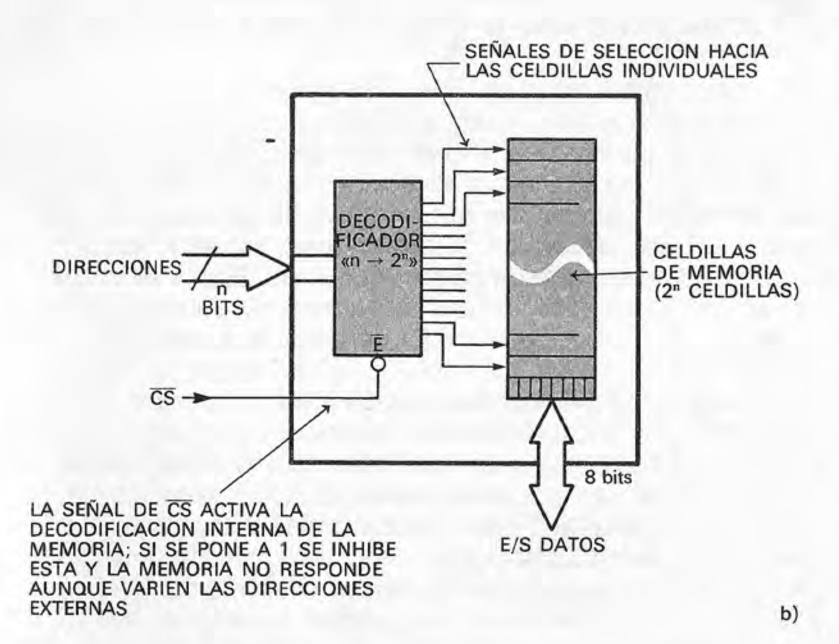




*Fig. 7 - (a) Un decodificador es como un selector mecánico cuyo cursor está posicionado, de forma unívoca, por la combinación aplicada a las líneas de entrada. La entrada de habilitación, si es inactiva (nivel lógico "1"), apaga el conmutador poniéndolo en la posición "no conectado" (NC); (b) estructura interna de una memoria RAM (o ROM).*



3 a 8 y 4 a 16, lo que muestra que las salidas son, en cualquier caso, igual a dos elevado al número de entradas ($4 = 2^2$, $8 = 2^3$ y $16 = 2^4$). Naturalmente, también el chip de memoria tiene en su interior un decodificador, aunque mucho más complejo, habida cuenta de que tiene que seleccionar una celda concreta entre millares. En el caso de una memoria RAM de 2K por ejemplo, la decodificación interna será del tipo 11 a 2048.

Como cada memoria tiene su propio decodificador habrá que conectar al chip tantas líneas de dirección (desde A0 en adelante) como entradas necesite el decodificador interno; las líneas no usadas del bus de direcciones se emplearán en los decodificadores externos.

Así para una memoria de M bytes harán falta: $2^m = M \rightarrow$ n (n.º líneas necesarias) = log M/log2. P.e. M = 2048 bytes → n = 11, tal como se vio en los ejemplos ilustrados.

En la figura 7b podemos ver el diagrama de bloques de un chip de memoria, con las celdillas y el decodificador interno para su selección. La señal "Chip Select" sirve, precisamente, para habilitar este este decodificador interno que, de no ser así, se mantendría inerte incluso cuando se activaran las "n" direcciones en su entrada.

En el ordenador de la figura 6 tenemos 48K de RAM, cuyos chips son reconocibles por el hecho de que tiene dos entradas de control RE y WE (activas en el nivel lógico "0"); otros 16K están destinados a la memoria ROM, que será materia a tratar en el apartado siguiente. ¿Les parece que una estructura de esta forma está ordenada y es sencilla? Pues bien, nos crea o no, salvo pocas modificaciones, también su ordenador personal tiene este tipo de organización. Todo cuanto le parezca misterioso, o complicado, en los esquemas incluidos en su manual, es sólo resultado de las características añadidas a esta base de partida común (por ejemplo, conexiones e interfaces diversas, discos, controlador de vídeo, coprocesadores, etc).

## ROM: "Ready Only Memory". La memoria con datos fijos

La memoria RAM tiene la peculiaridad de poder alterar el contenido de cada una de sus celdillas a voluntad de la

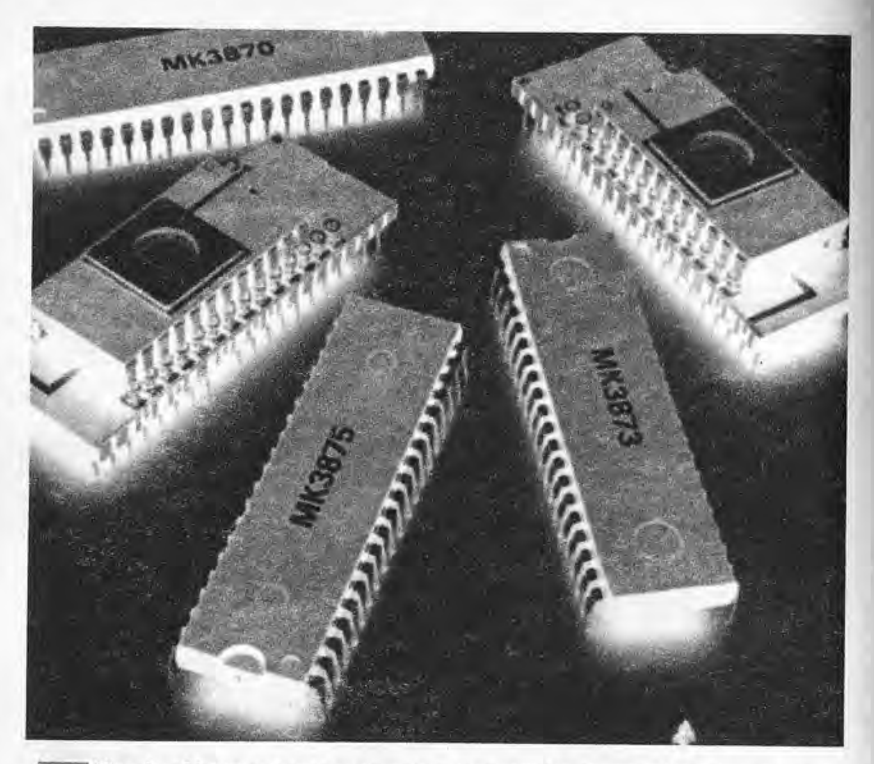



*Fig. 1 - Algunos de los circuitos integrados "monochips" más difundidos en el mercado. Cada cápsula contiene un único chip en el que están integrados, además de la CPU, una cierta cantidad de ROM, RAM y algunos dispositivos de E/S. El uso de estos circuitos de "un solo chip" está recomendado para todas aquellas aplicaciones en las que sea importante ahorrar espacio reduciendo el número de chips utilizados.*

CPU; por ello, el principal empleo de una memoria RAM es el almacenamiento provisional de datos (resultados intermedios, valores de variables, texto a procesar) y de programas. En resumen: de todo aquello que se puede rehacer, volver a escribir o cambiar. Existe solamente un "pequeño" inconveniente: una memoria RAM trabaja perfectamente mientras recibe energía, pero si cortamos la alimentación (apagamos el sistema) y luego lo volvemos a encender, nos encontraremos con una situación que podemos calificar de "desastrosa": el contenido anterior a la interrupción de corriente desaparecerá en la memoria RAM y será sustituido



por configuraciones totalmente aleatorias de códigos binarios de 8 bits que, por supuesto, no nos servirán para nada.

Conclusión: en una memoria RAM no pueden almacenarse datos de modo "permanente". ¿Cómo puede entonces nuestro ordenador personal "despertar", apenas encendido, y saber inmediatamente lo que debe hacer? Todo se debe a la existencia de las memorias ROM que, afortunadamente, están presentes en él. Son un tipo de memorias que no pierden los datos, aunque falte la alimentación. No obstante, como contrapartida, una memoria ROM no puede ser escrita como sucede con una RAM; es decir, puede ser leída, pero su contenido no puede modificarse mediante una operación de escritura normal.

El único modo de cambiar incluso un simple bit de una memoria ROM, es sustituirla por otra adecuadamente programada durante su fabricación. Es lamentable, ¡pero no se pueden pedir peras al olmo! La tecnología actual está invirtiendo mucho dinero en la búsqueda de dispositivos de memoria no volátiles que funcionen como la memoria RAM (es decir, que puedan ser escritas además de leídas) pero, por ahora, no se ha encontrado el sustituto ideal. Hay algunas "aproximaciones", como las memorias EEPROM y NOVRAM que funcionan bastante bien, pero sólo sirven para aplicaciones determinadas y, además, su coste es muy elevado. Las memorias CMOS, más que "no volátiles", lo que hacen es necesitar muy poca energía para conservar el dato, con lo cual una simple pila evita la pérdida de información.

En la práctica nuestro ordenador personal deberá presentar una mezcla adecuada de chips de memoria RAM y ROM, confiando a la primera el trabajo de almacenamiento provisional (con el ordenador personal encendido, para entendernos) y, a la segunda, la misión de retener aquellos datos necesarios para que arranque sin problemas el sistema en el momento del encendido y funcione correctamente después.

La estructura interna (tecnologías aparte) de una memoria ROM es idéntica a la de una memoria RAM, así como sus conexiones, con la salvedad de que en la ROM falta la línea de habilitación para la escritura puesto que, como es evidente, no serviría para nada.

## ¿Qué ocurre cuando conectamos el ordenador? Una nueva línea de control: RESET

Hasta que la tensión de la red no llega a la fuente de alimentación de nuestro ordenador, éste está mudo e inerte sobre la mesa; sin embargo su hardware está preparado para despertar de golpe apenas accionemos el interruptor.

La CPU, como sabemos, es una ejecutora obediente de instrucciones, que no son otra cosa que códigos de "unos" y "ceros" que extrae de la memoria sucesivamente. Cuando encendemos el sistema, un pequeño circuito exterior a la CPU pone, por unos instantes, al nivel lógico "cero" una entrada de la CPU, denominada de "Reset" (puesta a cero o inicialización). Un nivel lógico "cero" en esta entrada, es como el timbre que suena justo antes del comienzo de una clase: al oirlo, todos los alumnos permanecerán atentos esperando la llegada del profesor. En nuestro caso todos los dispositivos internos permanecen a la espera de comenzar a trabajar. Incluso el ID (codificador de instrucciones) queda esperando que la señal de Reset vuelva al nivel lógico "uno", después de lo cual, con su acostumbrada exactitud, comenzará a dirigir las tareas en el interior de la CPU.

Y he aquí que el Reset ha vuelto al nivel lógico "1". Al observar ese cambio el ID ordena al contador de programa que inicie la cuenta desde un valor determinado y característico de cada $\mu$P. Supongamos que este valor, naturalmente de 16 bits, sea el número binario 1111 1111 1111 1100 ($FFFC en hexadecimal). Dicha dirección, con la que el contador de programas se inicializará después del Reset se denomina "Vector de Reset". Pero el trabajo no está todavía terminado: el ID ordena inmediatamente que se lean los datos contenidos en esta dirección de Reset y en la inmediata superior. Estos dos valores de 8 bits son "capturados" por el ID, que los deposita en el contador de programa de forma que, finalmente, forman el valor de la dirección de memoria a partir de la cuál comienza el programa de inicialización propiamente dicho del ordenador. Es esa la razón del nombre "Vector de Reset", pues su contenido "apunta" al lugar donde comienzan las tareas de Reset.

Los más sagaces habrán sospechado que las dos direcciones consideradas (de reset) deben corresponder a un chip de ROM, pues es fundamental que su contenido sea siempre idéntico, para que en cualquier encendido el ordenador proceda de la misma forma.

Por las mismas razones las instrucciones del programa de inicialización deberán estar también en memoria ROM.

En pocos instantes la máquina estará completamente inicializada. A partir de ese momento será fundamentalmente la memoria RAM la que se ponga a disposición del usuario, quien podrá escribir sus programas, leerlos y cambiarlos utilizando la capacidad de almacenamiento de la memoria RAM. Pero si hay un "corte" en la alimentación ...¡adiós!

Ahora resulta más fácil comprender por qué razón existen los dispositivos de almacenamiento en soportes magnéticos (casete, disco): nos permiten copiar el contenido de la parte de RAM que nos interesa de forma que podamos apagar sin temor el ordenador y luego poder recuperar toda esa información.

En la figura 8 se ilustra la relación ROM/CPU durante la secuencia inmediatamente posterior a un Reset. Se supone que el denominado Vector de Reset está en $FFFE y $FFFF y que el programa de inicialización propiamente dicho está en memoria ROM, desde la dirección $F000 en adelante. El proceso será como sigue: fase 0) se enciende el sistema y a la CPU llega el impulso de Reset, pasando esta línea al nivel lógico "1"; fase 1) se situa el valor de la dirección "baja" de Reset ($FFFE); fase 2) se lee el dato contenido en esta

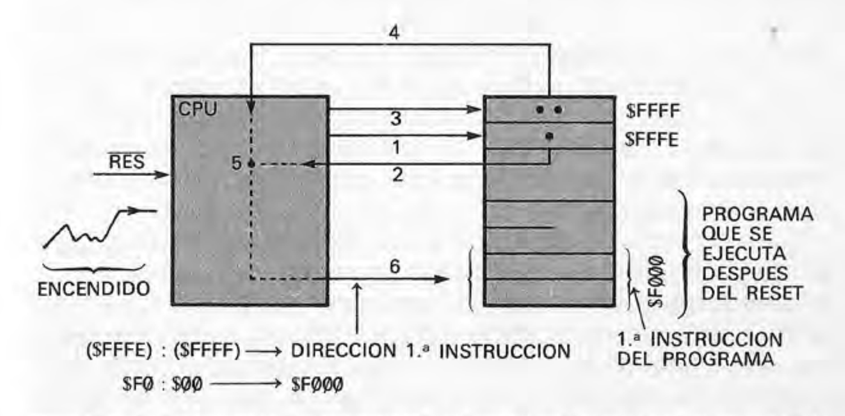



*Fig. 8 - Secuencia de reposición ("reset") subsiguiente al encendido del ordenador.*

dirección ($F0); fase 3) se emite el valor de la dirección "alta" de Reset y que equivale a la anterior +1; fase 4) se lee el contenido de la celdilla de memoria ROM correspondiente ($00); fase 5) en el interior de la CPU se ensamblan los dos datos de 8 bits en una dirección de 16 bits ($F000), que indica el verdadero comienzo, también en memoria ROM, del programa de inicialización; y fase 6) se pasa al contador de programa dicha dirección y se inicia la ejecución del programa de inicialización.

Para la explicación nos hemos basado en la secuencia de Reset de una conocida CPU (6809), pero cualquier otra sigue una secuencia similar, aunque las direcciones de Reset sean diferentes y sus contenidos también.

Esperamos que la función de una memoria ROM haya quedado clara. En el Glosario aportaremos información adicional sobre los diversos tipos disponibles en el mercado.

Ya tenemos una arquitectura basada en CPU, RAM y ROM, pero ¿sabemos cómo utilizarla de manera útil? Todavía no. Falta algo tan importante como para un automóvil pueden ser las ruedas: los dispositivos de entrada/salida.

## *Dispositivos de entrada/salida: la conexión con el mundo exterior*

Los dispositivos de entrada/salida (E/S) son circuitos a través de los cuales la CPU puede emitir o recibir señales hacia o desde el exterior. Dichas señales suelen ser de tipo binario porque son numerosísimos, en la práctica cotidiana, los dispositivos exteriores al ordenador que se pueden controlar, con facilidad, mediante señales del tipo "todo o nada". Así, por ejemplo, como dispositivos de salida podemos citar los relés, electro-válvulas y diodos LED; como dispositivos de entrada, los conmutadores, interruptores de fin de carrera, sensores, etc.

Para comprender cómo funciona y cómo está constituído un dispositivo de E/S, podemos extender la imagen que teníamos de nuestra conocida memoria RAM, con una única diferencia fundamental: cuando escribimos en una dirección perteneciente a un dispositivo de E/S, una circuitería particular del dispositivo (no presente en la memoria RAM normal) memoriza el contenido de la celda y lo reproduce en las líneas de señal correspondientes; estas son las que salen fuera de la cápsula y a las que se pueden conectar todos los dispositivos exteriores, tales como diodos LED, relés o cualquier otro que pueda ser controlado por señales digitales del tipo "todo o nada".



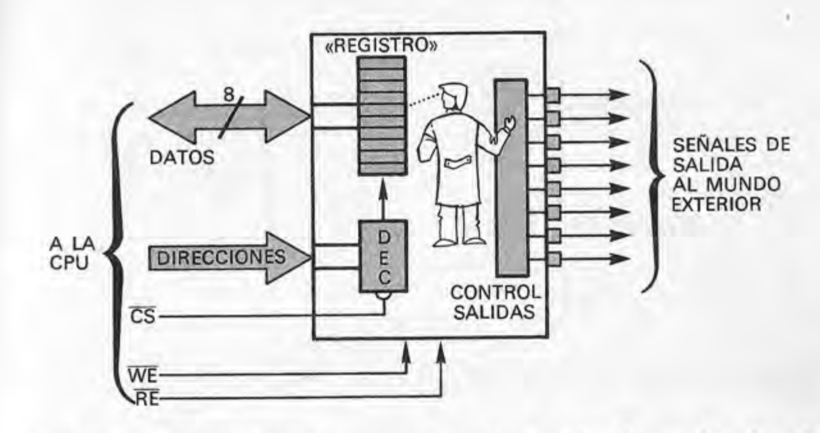



*Fig. 9 - Representación esquemática del interior de un chip de salida. El hombrecillo que copia el dato desde el registro a las líneas de salida en paralelo sintetiza las funciones realmente desarrolladas por una lógica digital sofisticada.*

En la Figura 9 se ilustra esquemáticamente el interior de un chip de salida con una única celdilla para contener los valores de las 8 señales de salida. Reconocemos el bus de datos, las líneas de direcciones que controlan el decodificador interior (habilitado a su vez por la señal Chip Select) y finalmente, la única celdilla existente que, al igual que una RAM de un solo byte, puede ser escrita y posteriormente leída (así podremos revisar lo que hayamos escrito). El hombrecillo simboliza la circuitería que extrae el contenido de la celdilla y lo reproduce en las patillas de salida, cuyo estado cambiará cada vez que la CPU escriba un nuevo dato en el chip.

La Figura 10 muestra un dispositivo de entrada, análogo al de salida visto con la salvedad de que ahora los terminales reciben señales (también de tipo "todo o nada") desde el mundo exterior. Cuando la CPU da orden de lectura al chip el hombrecillo copia el estado de las patillas en la única celdilla existente, y es el contenido de ésta el que pasa al bus de datos.

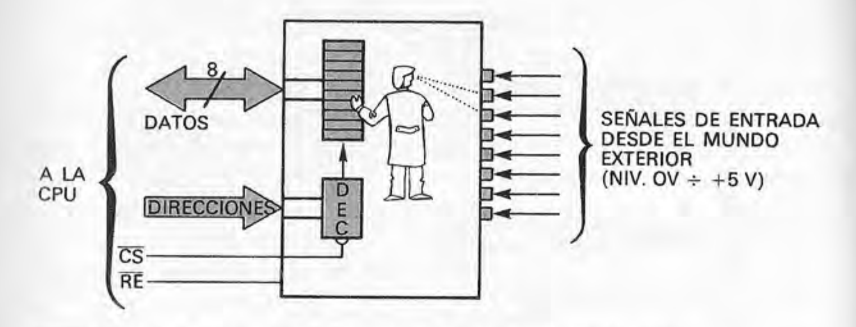



*Fig. 10 - Ilustración del interior de un chip de entrada.*

Las celdillas de los dispositivos de E/S suelen ser poco numerosas ya que a cada una corresponde, por lo general, un conjunto de 8 terminales ("Port") y la cápsula no suele tener más de 40. El nombre clásico de las celdillas de los chips de E/S es el de "registro"; así, para un dispositivo de salida se tendrá al menos un "registro de salida" y para un dispositivo de entrada habrá un "registro de entrada".

Es muy común encontrar dispositivos de entrada y salida en un solo chip. Entonces, el valor que introduzcamos en el registro "A" indicará cuál de las líneas es entrada y cuál salida según que su bit asociado sea 1 ó 0. En otra celdilla podremos escribir datos en las posiciones asociadas a líneas definidas como salidas o bien leer los datos existentes en las líneas definidas como entradas. El funcionamiento concreto, en todo caso, depende del tipo de dispositivo y del fabricante.

## Dos opciones: en paralelo o en serie

Por las descripciones del apartado anterior se puede deducir que los "ceros" y los "unos" del dato escrito en la salida (o leído en la entrada) llegan agrupados de 8 en 8 (en paralelo): una operación de escritura del dato binario 10101010, pondrá los ocho terminales de salida del circuito integrado respectivamente a las tensiones de +5V, 0V, +5V, etc., en una sola operación. Al "Port" (recordemos que Port = conjunto de 8 líneas de E/S) de salida podremos conectar simultáneamente 8 hilos controlando 8 dispositivos diferentes, o bien conectarlo al "Port" de entrada de otro ordenador y así las dos máquinas podrán comunicarse entre sí...



Sin embargo, manejar tantos cables puede resultar incómodo y caro, por lo que la aplicación de los dispositivos de E/S provistos de "Ports" en paralelo están limitadas a las transmisiones de señales a corta distancia y al control de sensores en aplicaciones industriales.

Más allá de los 3 ó 5 metros se prefieren normalmente los dispositivos de E/S en serie, que permiten la comunicación con un solo hilo (más la masa común, claro). De esta forma se limita el coste del cable de conexión (que se reduce a un cable apantallado) y al ser sólo uno podemos incluir la circuitería que nos garantice una buena protección frente a perturbaciones eléctricas, cosa que, de ser 8 o más los hilos, resultaría excesivamente costoso.

En la figura 11 se ilustra la forma en que se produce la transmisión en serie. En el dispositivo hay una celdilla (registro "TX"), en donde la CPU sólo puede escribir. Una vez terminada la escritura (que se realiza de forma análoga a la de una memoria RAM normal), nuestro hombrecillo accionará el interruptor "S", que controla el único terminal de sa-

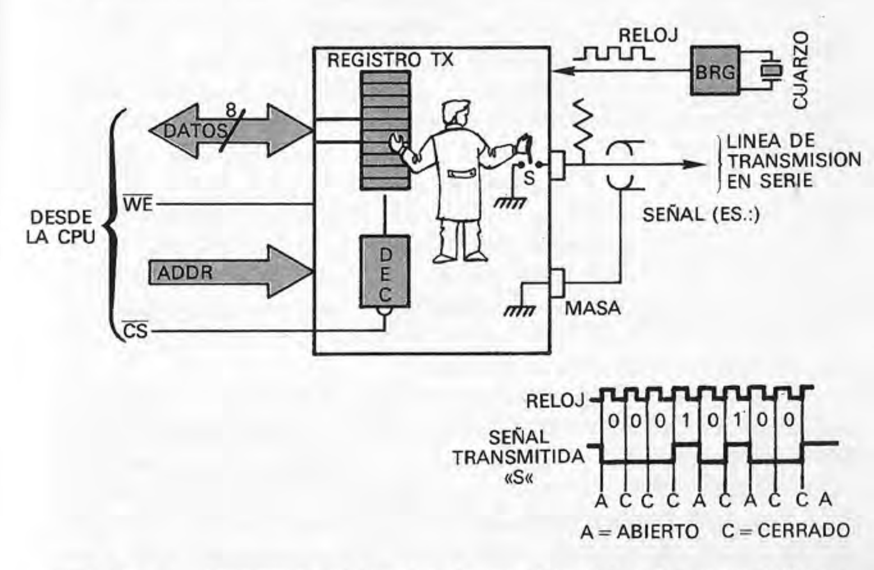



*Fig. 11 - Esquema del interior de un dispositivo de transmisión en serie.*

lida del circuito integrado. La existencia del "hombrecillo" se debe al deseo de evitar hablar innecesariamente por ahora de registros de desplazamiento, circuitos flip-flop, contadores, etc., circuitos todos ellos que se usan para lograr la transmisión, pero cuya mención sólo entorpecería la comprensión del proceso.

Volviendo al ejemplo, observará que si S está abierto, la línea va a "1" (hay una resistencia, llamada de pull-up=mantener a tensión, conectada a +), mientras que al cerrar S la línea va a "0". Por consiguiente, mediante la conmutación adecuada de S es posible generar formas de onda cuadrada a voluntad. La señal fundamental en los chips de E/S en serie y que no existe en los demás, es un reloj. Procede de un circuito exterior especial y consiste en un oscilador de cuarzo denominado "Baud Rate Generator" (generador de baudios o de velocidad de transmisión). Ahora bien, el baudio equivale a un bit por segundo (bps), lo cual significa que la velocidad (Baud rate) a que nos referimos es la de transmisión en serie del dato en el único hilo bit tras bit. La señal de reloj del generador obliga a que la lógica de control se sincronice con la máxima precisión, conmutando el interruptor S (cuando corresponda) solamente en correspondencia con las fases descendentes del propio reloj. Veamos un ejemplo para aclarar las cosas.

Si la CPU escribe en el registro TX el dato binario 00010100 ($14), el conmutador S deberá accionarse de modo que ponga la línea de salida a "cero" durante dos períodos del reloj, luego a "uno" durante un período, a "cero" durante otro período, a "uno" durante un período y, finalmente, a "cero" durante tres períodos. Después de ocho ciclos del reloj, el dato se habrá transmitido completamente a través de la única línea de salida. (Como se ha visto la transmisión comienza por el bit menos significativo, o de menos peso, que es el situado más a la derecha).

El dispositivo que puede recibir e interpretar de forma correcta un dato transmitido en serie, se representa de manera esquemática en la Figura 12. La señal entrante activa una lámpara que nuestro hombrecillo observa atentamente a la vez que comprueba el reloj de la velocidad de transmisión. Cuando hay un cambio en éste el hombrecillo apunta en el registro de recepción (RX) un 1 (lámpara encendida) o un 0 (apagada) en el bit correspondiente, llenando el



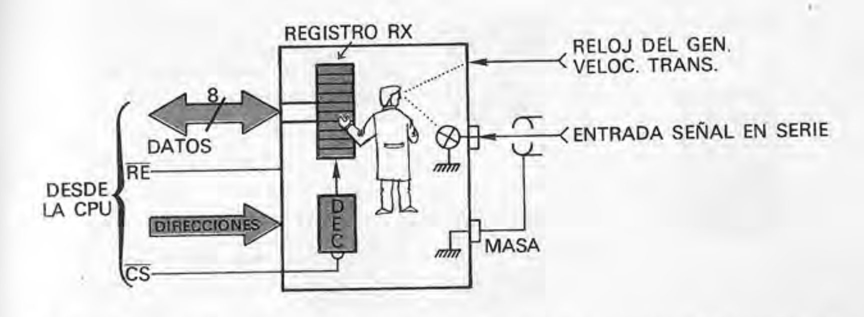



*Fig. 12 - Representación del interior de un dispositivo de recepción en serie.*

registro de derecha (bit menos significativo) a izquierda (más significativo).

Para que la interpretación del dato sea la correcta es preciso que el reloj del generador BRG sea idéntico para ambos dispositivos (emisor y receptor), de aquí la necesidad de emplear osciladores estables y, por ello, de cuarzo. A medida que se obtienen "unos" y "ceros" en la entrada, en sincronismo con el reloj, el registro de recepción RX se llena, como explicamos antes, comenzando por el bit 0 (el menos significativo) que, a su vez, se transmitió primero. Transcurridos ocho ciclos del reloj de transmisión/recepción el registro está completo y la CPU puede leerlo como una celdilla normal. Las velocidades de transmisión normales varían desde 300 a 9600 bits por segundo.

Algunos de estos dispositivos de E/S también incluyen varios integrados en el mismo chip, especialmente el generador de BRG, por lo que el diseñador sólo tiene que añadir el oscilador de cuarzo y conectar el chip a la CPU, sin más problemas.

## ¿Hace falta algo más? Del hardware al software

Si ha seguido hasta aquí sin problemas la lectura del libro podrá afirmar ya que conoce tanto el funcionamiento general de la CPU como el de los circuitos integrados de apoyo: RAM, ROM y E/S en serie y en paralelo. Sabrá valorar la

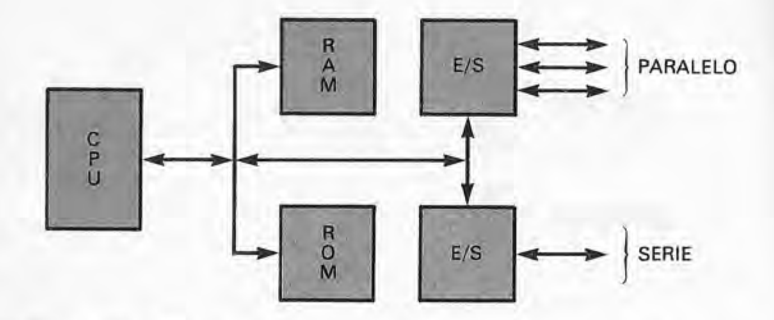



*Fig. 13 - Estructura completa de nuestro microordenador.*

importancia de las memorias ROM en la puesta en marcha del ordenador personal, la que tienen las memorias RAM en la retención temporal de datos y la de los dispositivos de E/S en la comunicación en ambos sentidos con el mundo exterior.

Ahora bien, todo este hardware, cuya descripción (a un primer nivel) consideramos concluída seguirá siendo una gran masa inerte de costoso silicio mientras no tenga un programa almacenado en el lugar oportuno de la memoria ROM. Efectivamente, para sacar provecho del hardware, es preciso disponer de un programa (software) que le indique lo que nosotros queremos que haga: por dónde tomar los datos, qué hacer con ellos, cómo presentarnos los resultados...

En los dos capítulos siguientes iremos viendo de qué forma será precisamente el software el que nos ayude a «personalizar» nuestro ordenador personal. Así que, si les parece, vayamos a ello, ¡traslademonos del hardware al software!

## GLOSARIO

KBYTES: Unidad de medida de la cantidad de memoria de un ordenador. Cada K (si no se dice lo contrario K se puede leer Kbytes) equivale a 1024 bytes (grupos contiguos de 8 bits). Una CPU normal de 8 bits, con un bus de direcciones de 16 líneas, puede direccionar un máximo de 64K (65.536 bytes).



HEXADECIMAL: Notación que se emplea al manejar datos binarios para no "marearnos" con tantos "ceros" y "unos". La notación hexadecimal hace uso de 16 dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F. Cada dígito hexadecimal representa una de las 16 combinaciones posibles con cuatro bits. Por ejemplo: 10100010 equivaldría al número hexadecimal A2 (normalmente se pone el signo $ delante —$A2— para indicar que se trata de un número hexadecimal). Para verlo, simplemente dividimos en grupos de 4 bits el número binario (comenzando por la derecha) y sustituímos cada uno por su equivalente hexadecimal. En el capítulo 5 daremos una descripción más detallada de este sistema de numeración.

DECODIFICADOR: Bloque lógico con un número de salidas que, como máximo, es la potencia de dos del número de entradas. Por ejemplo, si tiene 4 entradas, el máximo número de salidas será $2^4=16$. Su función es reconocer, de forma inequívoca, cualquier combinación de las entradas, activando la única salida correspondiente a dicha combinación. Los decodificadores se utilizan, fuera de la CPU, para obtener las señales de habilitación de los diversos chips de apoyo (RAM, ROM, etc.).

VOLATIL: Memoria que pierde todo su contenido cuando deja de aplicarse la alimentación. Típico ejemplo son las memorias RAM.

NO VOLATIL: Es un tipo de memoria como la RAM, pero que, en determinadas condiciones, e incluso sin alimentación, no pierde el contenido de sus celdillas (vea NOVRAM).

ROM: Abreviatura de Read Only Memory (Memoria de solo lectura). Nombre genérico de una memoria que sólo puede ser leída, no escrita, por el usuario. Su contenido se determina durante el proceso de fabricación mediante grabación "por máscaras".

PROM: Es como la anterior, con la salvedad de ser programable por el usuario mediante unas máquinas denominadas, lógicamente, "programadores de PROM". Una vez programada ya no puede modificarse su contenido y habremos de sustituirla por otra PROM si precisamos modificar el programa.

EPROM: Memoria PROM borrable (Erasable-PROM). Es como la PROM, pero se puede borrar exponiendo a cierta can-

tidad de rayos ultravioletas una mirilla transparente de la cápsula. Por consiguiente podemos reprogramarla. Es más cara que una ROM, pero permite un notable ahorro en la fase de puesta a punto y comprobación (con las modificaciones correspondientes) de un programa.

EEPROM: "Electrically-Erasable-PROM" (PROM borrable por medios eléctricos). De características similares a la EPROM, se puede borrar con bastante más comodidad que esta, ya que basta aplicar impulsos eléctricos en uno de sus terminales para borrar su contenido. Lamentablemente se trata de un chip de alto coste y su uso está limitado a ciertos fines específicos.

NOVRAM: "NOn-Volatile-RAM" (Memoria RAM no volátil). Se trata de una memoria RAM que contiene también una cantidad idéntica de EEPROM. Cuando cae la alimentación, el contenido de la memoria RAM se copia en la EEPROM y cuando vuelve la alimentación ocurre lo contrario: el contenido de la EEPROM se copia en la RAM y todo queda como antes. Son chips ideales contra las pérdidas de datos, pero su coste es muy elevado.

SISTEMA OPERATIVO: Programa original, incluído normalmente en las memorias ROM del ordenador por el fabricante, que controla el funcionamiento correcto del ordenador en sus relaciones con el usuario y con los periféricos.

VECTOR DE RESET: Se denomina "vector" a un par de posiciones contiguas de memoria cuyos valores forman otra dirección. En un $\mu$P el vector de Reset es el par de celdillas en las que la CPU buscará, inmediatamente después del Reset, la dirección efectiva en donde comienza el programa de inicialización.

REGISTRO: Nombre clásico de una celdilla genérica de un dispositivo de E/S o similar. Se aplica especialmente a celdillas muy específicas de la CPU o de los dispositivos de E/S (p. e. "registro de estado" en la CPU).

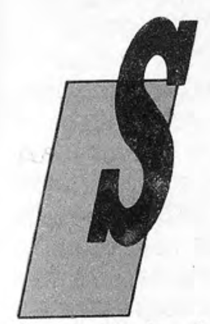
PORT: Nombre que se da al bloque de 8 terminales de entrada (port de entrada) o salida (port de salida) de un chip de E/S.

BAUDIO: Unidad de medida de la velocidad de una transmisión en serie; equivale a un bit por segundo.

SINGLE-CHIP (MICROCOMPUTER): Microordenador en un solo chip. Se trata de un circuito integrado que contiene una CPU, dispositivos de E/S y memoria RAM y ROM, todo ello integrado en el mismo chip de silicio. Se utiliza sobre todo en aplicaciones industriales en las cuales se construyan muchas unidades y que precisen una gran cantidad de memoria ROM (p. e. los microordenadores que controlan lavadoras, ascensores, etc.).

# CAPITULO IV

## SOFTWARE: EL COMBUSTIBLE DEL ORDENADOR



I amplio es el tema del hardware, tanto más lo es el del software: puede decirse incluso que, en cierta medida, el segundo es más importante que el primero, porque mientras las tecnologías constructivas mejoran de día en día, no es tan fácil conseguir expertos que sepan aprovechar al máximo toda la potencia de estas máquinas tan complejas. A título de ejemplo, emplear un mal programador con un ordenador bueno sería lo mismo que confiar a un automovilista normal la conducción de un Ferrari de Fórmula 1.

Los primeros ordenadores eran unas máquinas mostruosas, concebidas para facilitar lo más posible la realización de cálculos científicos complejos; su capacidad era comparable a la de algunas de las actuales calculadoras de bolsillo. El programa, es decir, la secuencia de operaciones a ejecutar, lo introducían por medios manuales técnicos especializados dado la complejidad del proceso. En resumen, era un ambiente reservado a unos pocos elegidos y, ciertamente, tanto en el fondo como en la forma, muy poco "user friendly" (amistoso con el usuario) como se dice en la actualidad, pues entre otras cosas se trabajaba casi constantemente en binario.

Más adelante, el célebre científico Von Neumann tuvo la brillante idea de considerar los códigos de operación de la máquina equivalentes, sino en el fondo al menos sí en la

forma, a los propios datos. Entonces se planteó el interrogante de por qué no podían estar ambos (códigos y datos) juntos en la memoria del ordenador "monstruo". De este modo, al ordenador se le proporcionaban de golpe, junto a los datos, la lista de las operaciones; luego se le dejaba libre para que prosiguiera de forma automática con la ejecución de los cálculos.

Aunque esto nos suena hoy como "los cuentos del abuelo", no hace mucho que ocurrió y nos muestra cuán difícil era, y sigue siendo, dialogar con una máquina que carezca del programa adecuado que la transforme de un monstruo inerte y estúpido en un colaborador siempre a nuestra disposición.

Sin profundizar demasiado en los conceptos de la programación (próximos volúmenes de esta colección lo harán), este libro quiere mostrar cómo es posible, gracias al software, hacer funcionar el hardware descrito anteriormente y conseguir finalmente una máquina dispuesta a prestar toda clase de servicios y con toda obediencia. Tomaremos, pues, como base, una arquitectura muy sencilla tal como la ilustrada en la Figura 1, con un dispositivo de entrada al que está conectado un teclado y un dispositivo de salida unido a una pantalla, además de una determinada cantidad de memoria RAM y ROM, por supuesto. En la ROM deberemos introducir el programa que hará funcionar al ordenador en el momento del encendido, de modo que podamos dialogar con el mismo permitiéndole comprender una sencilla instrucción de "?" (como la instrucción "PRINT" del BASIC), y que indique la existencia de un error si hubiéramos introducido el programa de modo equivocado. Es un ejemplo muy simple y con muchas limitaciones, evidentemente, pero servirá para hacernos comprender cómo, a partir del hardware y gracias al software, se llega a una máquina perfectamente "funcional".

Los próximos apartados serán indudablemente más difíciles de seguir para aquellos que no tengan experiencia alguna en programación, ni siquiera en BASIC, pero intentaremos que, incluso en esos casos, el texto resulte inteligible. De todas formas a quienes les queden dudas sobre esta materia podrán disiparlas consultando el resto de los volúmenes de esta obra o alguno de los libros citados en la Bibliografía final.





*Fig. 1 - El sencillo ordenador personal que estamos construyendo. Controla un teclado de 8 teclas y una pantalla. Puede comprender y ejecutar una sola única instrucción: <?> (PRINT).*

## El software del hardware

Perdone el juego de palabras, pero deseábamos insistir una vez más en que nuestro hardware debe poseer sus propios recursos de software, es decir, de programas. Esto significa que las instrucciones de los programas deberán estar en uno de los dispositivos de apoyo que hemos descrito en el capítulo anterior: en memoria ROM y RAM, cada una para cosas diferentes.

En la memoria ROM introduciremos el "software residente", denominado también "firmware", programado sobre ella en fábrica. El chip se colocará en la tarjeta a la vez que el resto del hardware del ordenador. En la memoria RAM, por el contrario, el usuario irá introduciendo su propio programa, una vez encendida la máquina y ejecutado el programa de inicialización.

Recordemos que programa es el nombre genérico dado a una secuencia de instrucciones, cada una de las cuales no

es otra cosa que un grupo de códigos binarios que pueden ser comprendidos por la CPU. Puesto que la CPU ejecuta las instrucciones una tras otra, el programa debe ser necesariamente compacto, es decir, los códigos de todas las instrucciones deben estar en celdillas contiguas en la memoria. En caso de que el programa tenga distintas partes separadas y quede "a trozos", al final de cada segmento de programa han de existir las oportunas instrucciones que digan a la CPU "no sigas en la dirección siguiente, ¡vete a la xyzw, donde está el próximo segmento del programa!". (Son las llamadas instrucciones "de salto").

Veamos de nuevo lo que sucede cuando conectamos el ordenador. En primer lugar se ejecuta la secuencia de «Reset», descrita en el capítulo anterior, lo que exige la presencia de, como mínimo, una memoria ROM con las primeras instrucciones de dicha secuencia. Y luego, ¿qué ocurre?

La respuesta es sencilla: la CPU prosigue la ejecución de otro programa, también residente en memoria ROM, que «personaliza» el hardware obligándolo a comunicarse con nosotros tal y como describía su manual de manejo. Carece de sentido explicar ahora, de forma detallada, las operaciones a realizar en una clásica secuencia de inicialización; bastará decir que suelen ser una serie de lecturas y escrituras en algunas zonas de la memoria y de los registros de los chips de E/S, con el objeto de que todos los puntos neurálgicos del hardware queden con valores conocidos, sobre los cuales se basan los futuros procesos. Veamos un ejemplo: en el capítulo anterior hablábamos de dispositivos con líneas que podrían ser entradas o salidas según el valor 1 ó 0 de su bit asociado dentro de un registro. Pues bien, si ese dispositivo lo usa nuestro ordenador, por ejemplo, para controlar un teclado y una pantalla, el programa de inicialización deberá cargar en ese registro los valores adecuados para que las líneas que, por hardware, hemos unido a la pantalla sean salidas y las conectadas al teclado sean entradas.

El hecho de que normalmente el ordenador admita datos introducidos a través de un teclado y visualice los resultados en una pantalla, nos da idea de la importancia de éstos; es tanta que muchas veces su control se realiza mediante chips de E/S especializados.

La Figura 2, por ejemplo, muestra la forma más simple de «leer» un teclado alfanumérico. En realidad, dicha reali-



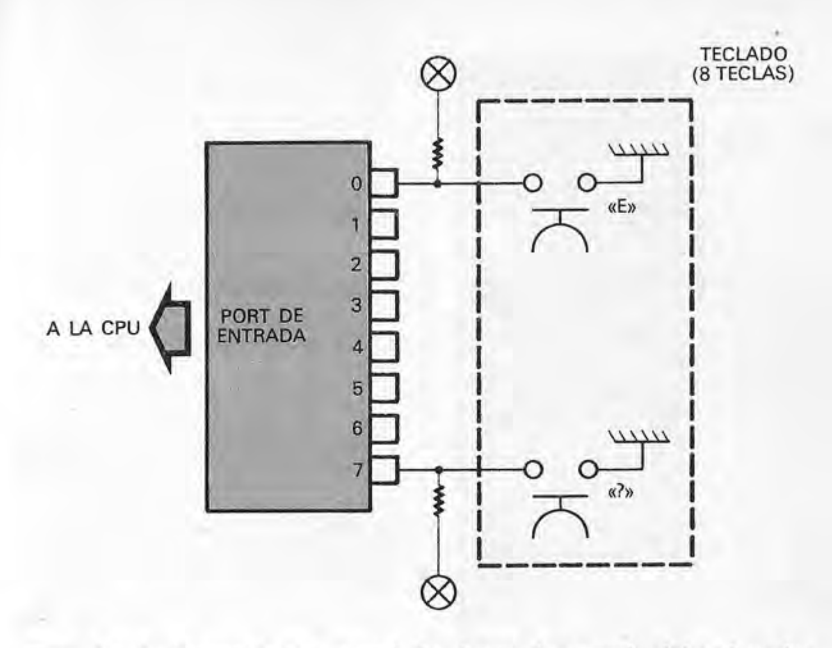



*Fig. 2 - Conexión de un teclado al port de entrada del ordenador.*

zación sería «despilfarradora»: nadie dedica una entrada individual del port de E/S para cada tecla, sino que se prefieren unas conexiones matriciales más complejas. No obstante, para hacer el ejemplo lo más sencillo posible admitiremos tener tantas entradas como teclas existen en el teclado, limitando este número a un total de 8 con miras a la máxima sencillez. Cada tecla es un pulsador que, presionado, une la entrada del port a masa; al soltar la tecla, la entrada volverá al nivel lógico «1» obligada por la visible resistencia de pull-up (puesta a tensión). Las entradas están agrupadas en grupos de 8, ya que se supone que cada port de entrada es de 8 bits.

Naturalmente, el chip de entrada al que está conectado el miniteclado debe estar en alguna parte en el mapa de memoria, puesto que la CPU ha de tener la posibilidad de leer el port efectuando una operación de lectura normal; por comodidad, la dirección del port para el teclado es $E000, en tanto será $E100, la dirección del port de salida del chip que controla la pantalla.

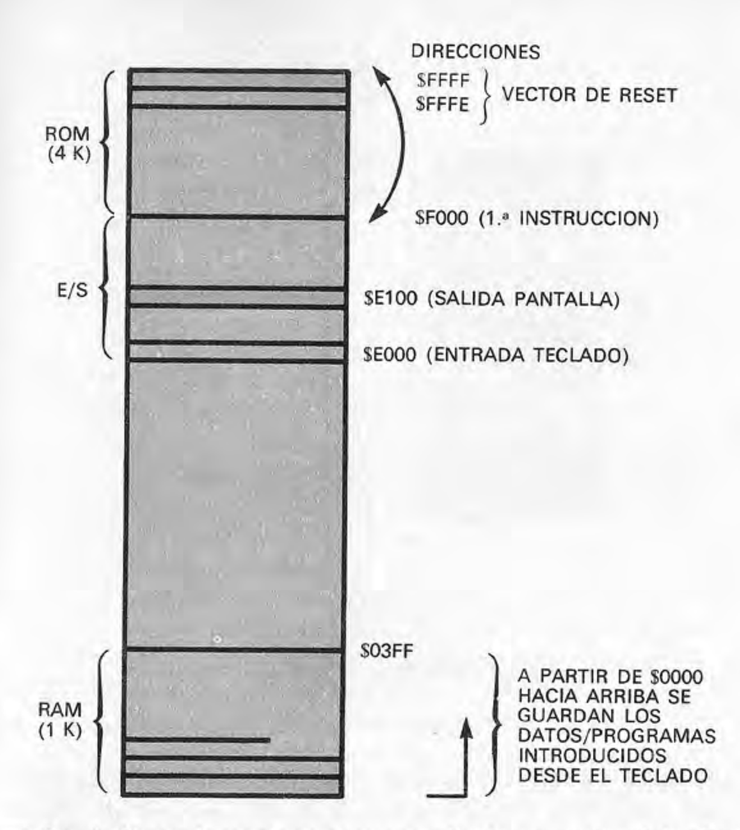



*Fig. 3 - Mapa de memoria correspondiente a nuestro miniordenador.*

La Figura 3 ilustra la disposición del mapa de memoria en nuestro ordenador. Se observa que la memoria RAM tiene una amplitud de 1.024 bytes (estamos con escasez de recursos...), desde $0000 a $03FF, mientras que más arriba están los chips de E/S y luego, desde $F000 hasta el final —es decir $FFFF—, hay 4.096 bytes de memoria ROM (más que suficiente, como se verá más adelante).

Volviendo a nuestro pequeño teclado, en la Tabla 1 se indican los nombres de sus teclas y el código resultante en los terminales del port cuando se pulsan, tanto en forma binaria como hexadecimal.

Probemos ahora a imaginarnos cuál puede ser la serie de instrucciones que permite a la CPU reconocer la tecla que hayamos pulsado. Hemos de tener presente qué es,



| TECLA | CODIGO BINARIO | CODIGO HEXADECIMAL |
|---|---|---|
| E | 1 1 1 1 1 1 1 0 | F E |
| R | 1 1 1 1 1 1 0 1 | F D |
| O | 1 1 1 1 1 0 1 1 | F B |
| A | 1 1 1 1 0 1 1 1 | F 7 |
| B | 1 1 1 0 1 1 1 1 | E F |
| " | 1 1 0 1 1 1 1 1 | D F |
| «CR» | 1 0 1 1 1 1 1 1 | B F |
| ? | 0 1 1 1 1 1 1 1 | 7 F |

*Tabla 1 - Codificación de las teclas del teclado.*

realmente, una operación de lectura: la CPU emite una dirección y solicita al dispositivo que se encuentra en esa dirección (ROM, RAM o bien E/S, como en este caso) que transmita el dato de 8 bits que contiene. La sentencia:

LDA $E000

indica, en el código «nemónico» que resume la instrucción, una operación de lectura del contenido de una cierta dirección ($E000, en este caso). Los «nemónicos» son las instrucciones de un lenguaje llamado Ensamblador, cuya única finalidad es establecer una correspondencia directa y única de cada código máquina con un «nemónico», es decir, con una palabra que nos recuerda qué es lo que hace ese código con el que va asociado; de esta forma el usuario puede escribir primero el programa en Ensamblador, sin tener que memorizar el significado de cada dódigo y luego traducir el programa así escrito a los correspondientes códigos máquina. (Vea el Glosario para más información).

Por supuesto, sabemos que el decodificador de instrucciones (ID) que existe dentro de la CPU y que tiene como función supervisar la ejecución de las instrucciones, no comprende la instrucción «LDA», sino solamente códigos binarios bien precisos. Ante esta circunstancia, tomaremos el manual de la CPU y, buscando el código que corresponde a la instrucción LDA, encontramos, por ejemplo, 10101101 (o bien $AD). Escribimos entonces la instrucción, y el progra-

ma que al final introduciremos en memoria ROM se va convirtiendo en:

INICIALIZACION (secuencia de códigos que pasamos por alto)

AD E0 00

Una vez leído el Port, es preciso comprobar si hay una tecla pulsada; esta operación es fácil, puesto que si no la hay el dato leído será $FF, es decir, 11111111 en binario. Si, por el contrario, hay alguna tecla pulsada, habrá un «cero» en una posición cualquiera de las ocho leídas; entonces, la solución más sencilla y rápida consiste en admitir como código de la tecla el valor leído (según se indica en la Tabla 1). Para simplificar consideramos a priori que no es posible pulsar simultáneamente varias teclas, por lo que estamos seguros de que los códigos leídos son verdaderamente los correspondientes a las teclas pulsadas.

La comprobación del valor leído por la CPU es fácilmente realizable mediante una instrucción de comparación del tipo «CMPA #$FF», que significa «compara el dato que acabas de leer (en la instrucción anterior) y que tienes en el acumulador con el valor que sigue, es decir, $FF (11111111 en binario). Volvemos a consultar el manual de la CPU y encontramos que esto equivale para la CPU a:

C9 FF

y, por consiguiente, nuestro programa se convierte ahora en:

INICIALIZACION
AD E0 00
C9 FF

Después de la comparación es preciso ver cuál ha sido el resultado. En nuestro caso, debiéramos volver a intentar una lectura si no encontramos ninguna tecla pulsada, es decir, si el dato leído era igual a $FF. La instrucción puede ser «JEQ-5», o bien «Si, y solamente si, el valor leído es igual (EQual) al de comparación, salta (Jump) cinco pasos hacia atrás (–5).» El resultado es que la CPU se vería obligada a repetir la instrucción de lectura que se inicia con «AD».



De este modo, hemos creado un «bucle» o ciclo en nuestro programa. Se repetirá indefinidamente hasta que pulsemos una tecla. El programa es ahora:

INICIALIZACION
AD E0 00 (Carga el valor de E000)
C9 FF (Lo compara con $FF)
F0 –5 (Si es igual salta de nuevo a $AD. Nos reservamos para más adelante la traducción en binario del –5)

Como puede observar, con solamente tres instrucciones hemos traducido a códigos interpretables por la CPU este razonamiento: «sigo leyendo la dirección del Port de entrada, al que está conectado el teclado, hasta que encuentre una tecla pulsada». ¿Es posible, prosiguiendo con este modo de proceder que seguimos: RAZONAMIENTO → OPERACIONES NECESARIAS → CODIFICACION BINARIA DE LAS INSTRUCCIONES desarrollar todo el programa en la forma requerida?

Desde luego que sí, pero también es alta la probabilidad de introducir errores, porque se pierde de vista la «lógica» que el programa debe seguir. Ha llegado el momento, antes de completar nuestra obra, de que tratemos de analizar la Figura 4 y comprender su utilidad.

## *Diagramas de flujo*

Como muchos de nuestros lectores posiblemente sabrán, un diagrama de flujo es una representación gráfica que permite poner de manifiesto, con los símbolos adecuados, la sucesión de operaciones necesarias en un programa, simplificando su comprensión y facilitando las tareas de conservación y modificación posteriores.

La Figura 4 ilustra un sencillísimo diagrama de flujo, relativo al miniprograma que estamos escribiendo. Observará que en una sola figura está condensada toda la serie de operaciones necesarias para el buen funcionamiento del programa, descritas sin que aparezca ninguna intrucción en código máquina (pues no pretendemos en absoluto utilizar este esquema para codificar las instrucciones en el lengua-

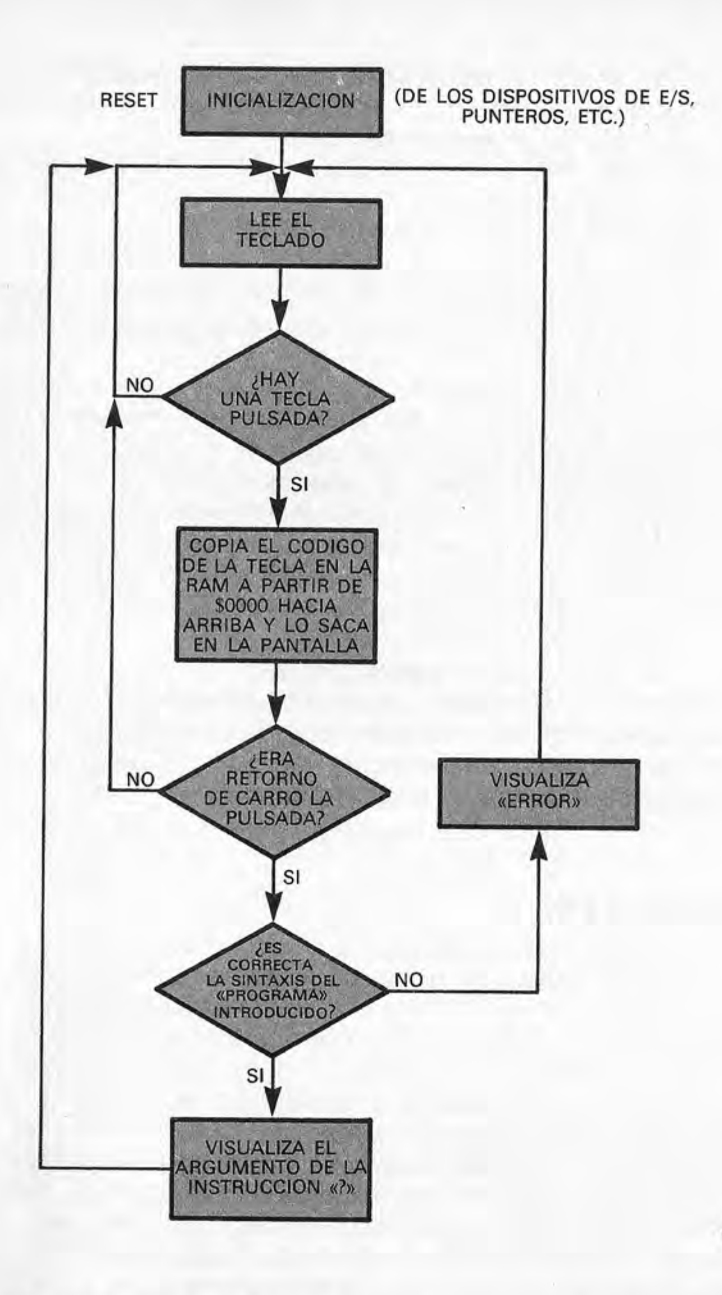



*Fig. 4 - Diagrama de flujo del programa que controla nuestro miniordenador.*



je máquina de una CPU específica). Probemos a leer el diagrama.

Se inicia con un rectángulo, un símbolo frecuentemente utilizado para reagrupar una serie de instrucciones que han de ejecutarse de forma secuencial. De hecho, el primer rectángulo simboliza la totalidad de las instrucciones preliminares dedicadas a la inicialización de la máquina, situación que hemos tomado como punto de partida.

La lectura prosigue en el sentido de las flechas y se llega al segundo rectángulo, que indica una operación de lectura, precisamente del teclado.

La tercera figura es un rombo, símbolo «de decisión» que indica una «prueba»; es decir, la comprobación de un valor. El bloque tiene dos salidas: una si la prueba ha dado resultado afirmativo y la otra en caso contrario. En nuestro ejemplo, hasta que no haya alguna tecla pulsada el camino a seguir es siempre el de la flecha que sale por «no», por lo que volveremos al rectángulo que indica la lectura del teclado. Es muy clara la existencia del anillo de repetición («bucle») al que hemos hecho referencia mientras escribíamos las primeras instrucciones del programa.

¿Qué sucede si pulsamos una tecla? Continuando con la interpretación del diagrama, se observa que el código de la tecla se copia en la memoria RAM, a partir de la dirección $0000 hacia arriba, y se reproduce en la pantalla. Después el programa solicita una segunda prueba para comprobar si la tecla pulsada era un «retorno de carro» («Return»). Si no lo era, la salida «No» del bloque de prueba vuelve a llevarnos a una nueva lectura del teclado, con lo que se crea un segundo bucle que envuelve al anterior, más pequeño.

Si la tecla pulsada coincide con el retorno de carro, se considera que el usuario quiere indicar que ha acabado de introducir su programa y, por ello, el bucle se termina, ejecutándose el bloque siguiente. Pero antes de continuar, podemos tener la curiosidad de saber a dónde irá a parar el programa escrito por el usuario y cómo llegó allí.

Pues bien, volvamos a analizar el comportamiento de los dos bucles anteriores; para cada pulsación de una tecla, incluyendo la de retorno de carro, los códigos correspondientes se guardarán sucesivamente en la memoria RAM a partir de la dirección $0000 en adelante. El programa intro-

ducido por el usuario se leerá posteriormente en esta parte de la memoria del sistema, y estará constituido por una secuencia de códigos que no son otros que los que hemos atribuido arbitrariamente a las 8 teclas (vea Tabla 1).

El programa escrito por el usuario puede estar constituido por códigos que no tengan sentido para la CPU. Para que puedan ser comprendidos por dicha unidad es preciso que ella misma esté ejecutando otro programa (más «profundo») que le sirva de intérprete entre los códigos empleados por el usuario y los que la CPU maneja.

En otros términos, esto significa que deberemos escribir, utilizando el código máquina propio de la CPU, un programa «intérprete» gracias al cuál podremos escribir otros programas cuyas instrucciones utilizarán códigos completamente diferentes a los de la máquina. La razón de esto es simplemente poder usar códigos que «nos digan algo» sobre la operación a realizar, cosa que no ocurre con los que emplea la máquina (AD, C9, F0,...)

Dichos programas «intérpretes» se denominan, por lo general, «lenguajes». Se establece así un «doble nivel» en la ejecución de los programas que escribimos cuando trabajamos con nuestro ordenador personal. Por ejemplo, si nosotros programamos empleando un lenguaje sencillo, el BASIC, cuyas instrucciones están constituidas por palabras fáciles de escribir y recordar, este programa, escrito en BASIC, e introducido en la memoria del ordenador personal a través del teclado, es «interpretado» por otro programa residente en el ordenador, a menudo en una memoria ROM producida por el mismo fabricante del ordenador personal (en otros casos, se carga en memoria RAM desde el disco o la cinta. El resultado de esta interpretación es, finalmente, la secuencia de instrucciones en código máquina que sí son ejecutables por la CPU.

## *Los últimos toques*

La Figura 5 muestra el interface de video que necesitamos. Lo trataremos como un circuito integrado normal de E/S. Más adelante profundizaremos en el funcionamiento de los terminales, pero por ahora nos contentaremos son saber que la CPU, para presentar un carácter en la pantalla, sim-



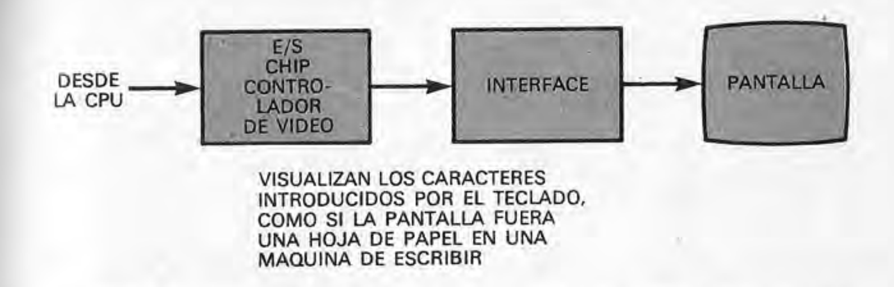


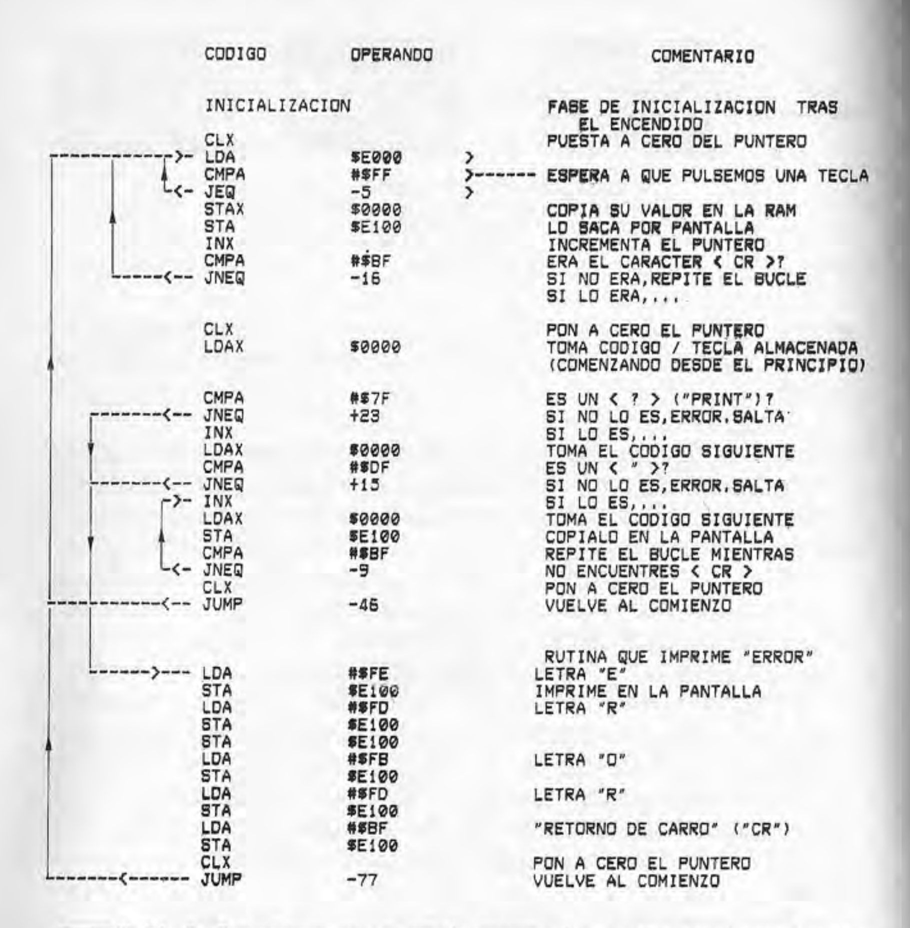

*Fig. 5 - Conexión de la pantalla, manejada por un chip controlador de vídeo especial que, ante la CPU, aparece como un port de salida normal.*

plemente efectúa una operación de escritura en un registro concreto del chip de E/S dedicado al control de la pantalla. Las complejas funciones internas del chip de control supondremos se preestablecieron durante la fase de inicialización.

Si ahora revisamos en el diagrama de flujo las operaciones realizadas por el bloque señalizado con el asterisco observamos que, apenas se ha detectado la pulsación de una tecla, el código que la describe no solamente se copia en otra parte de la memoria, sino que se introduce en el chip de video; de esta manera tendremos inmeditamente la visualización de todas las teclas pulsadas. Cuando pulsamos la tecla «Return» (CR, Retorno del carro) indicamos que hemos acabado la introducción de códigos, por lo que la máquina puede comenzar a traducir (si puede) las órdenes impartidas.

En nuestro caso definiremos una sola instrucción, la «?», para la cual establecemos que la sintaxis válida es la siguiente: <?"xyzw..wxz(CR)>. Dicho de otro modo, después de la tecla <?> ha de pulsarse la tecla <">, que indica el comienzo de la cadena alfanumérica que deberá escribirse en la pantalla. La cadena termina con el <CR>, es decir, con el retorno de carro o «return» del teclado. Cualquier otra combinación de pulsaciones la consideraremos no válida, y el ordenador deberá indicarlo con la presencia visual del mensaje <ERROR>.

Nuestro microsistema operativo es muy sencillo y no tiene en cuenta muchas de las situaciones que podrían tener lugar, por lo que podríamos decir que no está todavía «depurado». No obstante, tampoco pretendemos llegar más



```
             CODIGO        OPERANDO                 COMENTARIO

             INICIALIZACION                         FASE DE INICIALIZACION  TRAS
                                                       EL ENCENDIDO
             CLX                                    PUESTA A CERO DEL PUNTERO
-------->-   LDA           $E000        >
             CMPA          #$FF         >------ ESPERA A QUE PULSEMOS UNA TECLA
      <-     JEQ           -5           >
             STAX          $0000                    COPIA SU VALOR EN LA RAM
             STA           $E100                    LO SACA POR PANTALLA
             INX                                    INCREMENTA EL PUNTERO
             CMPA          #$BF                     ERA EL CARACTER < CR >?
  -----<--   JNEQ          -16                      SI NO ERA,REPITE EL BUCLE
                                                    SI LO ERA,...

             CLX                                    PON A CERO EL PUNTERO
             LDAX          $0000                    TOMA CODIGO / TECLA ALMACENADA
                                                    (COMENZANDO DESDE EL PRINCIPIO)

             CMPA          #$7F                     ES UN < ? > ("PRINT")?
 -------<--  JNEQ          +23                      SI NO LO ES,ERROR.SALTA
             INX                                    SI LO ES,...
             LDAX          $0000                    TOMA EL CODIGO SIGUIENTE
             CMPA          #$DF                     ES UN < " >?
 -------<--  JNEQ          +15                      SI NO LO ES,ERROR.SALTA
        ->-  INX                                    SI LO ES,...
             LDAX          $0000                    TOMA EL CODIGO SIGUIENTE
             STA           $E100                    COPIALO EN LA PANTALLA
             CMPA          #$BF                     REPITE EL BUCLE MIENTRAS
         <-  JNEQ          -9                       NO ENCUENTRES < CR >
             CLX                                    PON A CERO EL PUNTERO
 ---------<-- JUMP         -46                      VUELVE AL COMIENZO

                                                    RUTINA QUE IMPRIME "ERROR"
 ------>---  LDA           #$FE                     LETRA "E"
             STA           $E100                    IMPRIME EN LA PANTALLA
             LDA           #$FD                     LETRA "R"
             STA           $E100
             STA           $E100
             LDA           #$FB                     LETRA "O"
             STA           $E100
             LDA           #$FD                     LETRA "R"
             STA           $E100
             LDA           #$BF                     "RETORNO DE CARRO" ("CR")
             STA           $E100
             CLX                                    PON A CERO EL PUNTERO
 -------<------ JUMP       -77                      VUELVE AL COMIENZO
```

*Tabla 2 - Programa que controla nuestro hardware, escrito en los nemónicos del lenguaje ensamblador de una CPU imaginaria (que se parece un poco a las 6502, 6809, etc.). Junto a las instrucciones está el comentario correspondiente.*

allá de lo que es el ejemplo por lo que consideraremos el diagrama de flujo y su traducción al lenguaje máquina (Tabla 2) tal como está, sin meternos en complicaciones.

En la Tabla 2 hemos completado la escritura de nuestro software residente, utilizando un lenguaje máquina «semi-inventado», con el objetivo de dar a las instrucciones nombres fáciles de comprender. Probemos a leerlo.

La fase de INICIALIZACION se examinó anteriormente.



El «verdadero» programa se inicia con la puesta a cero de un puntero (instrucción CLX, que se verá dentro de poco). Encontramos luego el comienzo del primer bucle, dedicado a vigilar el teclado hasta que se pulse una tecla. Cuando se pulse, el bucle se interrumpirá y el código de la tecla se almacenará en memoria, «a partir de la posición $0000 hacia arriba».

Esta operación se ejecuta con una instrucción de «escritura con puntero». El puntero (X) es otro registro interno de la CPU (normalmente existe más de uno), que puede ponerse a cero e incrementarse de 1 en 1 con una instrucción INX (INcrementa el puntero X). La instrucción de escritura con puntero, la «STAX», comunica a la CPU lo siguiente: «guarda el valor que tengas en el acumulador en la cedilla de memoria cuya dirección viene dada al sumar el valor que te doy a continuación —dirección «base»— con el contenido del registro puntero «X».

En nuestro programa, vemos que la dirección «base» es siempre $0000, por lo que si X es 0, como ocurre después de una instrucción de puesta a cero (CLX), la intrucción STAX $0000 equivale a STA $0000; si, por el contrario, X con tiene otro valor (1, 2, 3, etc.), la misma instrucción STAX $0000 equivaldrá respectivamente a: STA $0001, STA $0002, STA $0003, etc. De este modo, se puede simplificar notablemente la escritura de un programa. Ahora podemos comprender para qué servía la instrucción CLX situada al comienzo.

Después de copiar el código en la dirección de RAM correspondiente (en este primer momento sería $0000) lo copia en $E100 para sacarlo a pantalla. Después hace INX para elevar en 1 el contenido de X (así ahora STAX $0000 equivaldrá a STA $0001, con lo cual no borro el dato que antes guardé en $0000).

Este proceso se irá repitiendo mientras pulsemos cualquier tecla que no sea un retorno de carro <CR>. Este hecho es comprobado en la instrucción CMPA #$BF. Dicha comparación va seguida por un salto condicional, que produce la repetición del bucle si la tecla no fuera un retorno de carro, pues si el código guardado en el acumulador no es $BF salta 16 bytes hacia arriba en el programa.

Si llega dicho retorno de carro pone de nuevo a cero el puntero (CLX) y luego comenzará a leer el contenido de

la memoria RAM a partir de $0000 hacia arriba (instrucción LDAX $0000; equivalente en su mecanismo a la STAX, pero en modo lectura), comprobando si la sintaxis era correcta. Esta es la fase de interpretación, propiamente dicha, del programa introducido por el teclado.

Según lo previsto deberemos encontrar en primer lugar el código de la única instrucción admitida, la <?>, que es $7F. Si no fuera así, habremos cometido un error y se producirá el salto JNEQ, que significa «salta (Jump) si los valores comparados antes No son iguales (EQual)». Aquí vemos que se salta 23 pasos hacia adelante. Si el resultado de la prueba fuera positivo (hay un $7F) se incrementaría el puntero para poder tener acceso a la dirección siguiente ($0001) utilizando también la instrucción LDAX $0000. El valor de este segundo código debe ser <">, que indica el comienzo de la cadena a visualizar. Otra prueba, y otro salto JNEQ, si no es el código requerido.

Por contra, si todo es correcto, se iniciará un nuevo bucle que tiene por objeto pasar al chip de E/S correspondiente a la presentación visual, toda la secuencia de códigos almacenados en RAM, que se encuentran después de los dos primeros códigos de instrucción, para su visualización en la pantalla. El bucle termina cuando se encuentra el código de retorno de carro (<CR>), después de lo cual se pone a cero el puntero y el programa se reinicia desde el comienzo (observe que hubiera sido lo mismo no poner el CLX, pues en ese caso JUMP-46 produciría un salto al primer CLX del programa). La máquina, por consiguiente, quedará a la espera de un nuevo ciclo de pulsaciones de teclas.

Las últimas líneas del programa proceden a la visualización del mensaje «ERROR», al término del cual se volverá al comienzo con el puntero puesto a cero.

¿La parece complicado? Pruebe a comparar la Figura 4 (diagrama de flujo) y la Tabla 2 (programa escrito en lenguaje Ensamblador). ¿A que verdaderamente, no son tan diferentes?

Hasta ahora no hemos hecho otra cosa que poner en práctica el siguiente procedimiento:

1) Tener las ideas claras sobre lo que se le exige al hardware incorporado.



2) Desarrollar un memorándum del funcionamiento global de la máquina.
3) Realizar el diagrama de flujo de las operaciones a efectuar.
4) Traducir el diagrama a una secuencia de instrucciones escritas en el lenguaje ensamblador de una supuesta CPU.

Los pasos que nos quedan son:

5) Codificar el programa, instrucción por instrucción, para obtener la secuencia de códigos máquina (binarios), ejecutables por la CPU.
6) Programar una memoria ROM con los códigos antes citados, de forma que constituirá nuestro programa residente (firmware) e insertarla en su zócalo dentro de la placa realizada.
7) Encender la máquina y verificar que el programa funciona de forma correcta; de no ser así, será preciso encontrar los errores y repetir el proceso desde el punto 3.

Llevémoslos a cabo.

Mediante el manual de programación de la CPU utilizada podemos traducir cada instrucción de la Tabla 2 en su código binario correspondientes (hexadecimal, para tener más comodidad en la escritura); como resultado obtendremos algo «parecido» (todo dependerá de cuáles sean los códigos de nuestra CPU) a lo expuesto en la Tabla 3. Para que siga la «conversión», al final de la tabla le indicamos las correspondencias que marcaba nuestro «inexistente» manual. Entre otras cosas, los valores negativos y positivos de los saltos (instrucciones JUMP) se han convertido en códigos hexadecimales (cómo es una cuestión a la que responderemos en el capítulo siguiente).

El programa habrá de grabarse (vea el mapa de memoria de la Figura 3) desde la dirección $F000 en adelante incluyendo la inicialización. En la Tabla 3 hemos supuesto que ésta ocupa un espacio de $89 bytes (137 en decimal), desde $F000 a $F088, por lo que la primera instrucción de nuestro programa «de verdad» (CLX) se escribe en la dirección $F089; como ocupa un byte de memoria la segunda

| CODIGO HEXADECIMAL COD.OP. | OPERANDO | | DIRECCION DEL PRIMER BYTE DE LA INSTRUCCION |
|---|---|---|---|
| INICIALIZACION | | | DESDE $F000 A $F088 |
| E9 | | | $F089 |
| AD | E0 00 | | $F08A |
| C9 | FF | | $F08D |
| F0 | FB | (-5) | $F08F |
| 9D | 00 00 | | $F091 |
| 8D | E1 00 | | $F094 |
| E8 | | | $F097 |
| C9 | BF | | $F098 |
| D0 | F0 | (-16) | $F09A |
| E9 | | | $F09C |
| BD | 00 00 | | $F09D |
| C9 | 7F | | $F0A0 |
| D0 | 17 | (+23) | $F0A2 |
| E8 | | | $F0A4 |
| BD | 00 00 | | $F0A5 |
| C9 | DF | | $F0A8 |
| D0 | 0F | (+15) | $F0AA |
| E8 | | | $F0AC |
| BD | 00 00 | | $F0AD |
| 8D | E1 00 | | $F0B0 |
| C9 | BF | | $F0B3 |
| D0 | F7 | (-9) | $F0B5 |
| E9 | | | $F0B7 |
| 4C | D2 | (-46) | $F0B8 |
| A9 | FE | | $F0BA |
| 8D | E1 00 | | $F0BC |
| A9 | FD | | $F0BF |
| 8D | E1 00 | | $F0C1 |
| 8D | E1 00 | | $F0C4 |
| A9 | FB | | $F0C7 |
| 8D | E1 00 | | $F0C9 |
| A9 | FD | | $F0CC |
| 8D | E1 00 | | $F0CE |
| A9 | BF | | $F0D1 |
| 8D | E1 00 | | $F0D3 |
| E9 | | | $F0D6 |
| 4C | B3 | (-77) | $F0D7 |

DIRECCION DEL ULTIMO BYTE: $F0D8

*Tabla 3 - Codificación, en hexadecimal, del programa escrito en la tabla 2. Se observa el carácter biunívoco entre la versión nemónica y la codificación hexadecimal de cada instrucción. A la derecha, se indica la dirección de memoria donde está situado el primer byte (el código operativo) de cada instrucción. Para realizar la codificación, hemos usado un ficticio manual de programación de nuestra "inventada" CPU, en donde estarían indicadas las correspondencias siguientes (nemónico - código máquina en hexadecimal):*

| | | | | |
|---|---|---|---|---|
| *CLX* | *E9* | | *STA* | *8D* |
| *LDA* | *AD* | *(DESDE LA MEMORIA)* | *INX* | *E8* |
| *LDA* | *A9* | *(INMEDIATO)* | *JNEQ* | *D0* |
| *CMPA* | *C9* | | *LDAX* | *BD* |
| *JEQ* | *F0* | | *JUMP* | *4C* |
| *STAX* | *9D* | | | |



instrucción (LDA $E000) se escribirá en la dirección $F08A y al ocupar tres bytes, la tercera instrucción se escribirá (dos bytes) en $F08D y en $F08E, la cuarta en $F08F y $F090, y así sucesivamente. En la misma tabla, junto a cada instrucción codificada, se indica la dirección en donde guardamos el primer byte de la instrucción correspondiente. La longitud total del programa es de $D9 bytes (en decimal 217), desde $F000 a $F0D8.

Para ejecutar el paso 6, se utiliza una máquina especial denominada «Programador de PROM», y se programa una PROM (o una EPROM o EEPROM), con los códigos antes citados. Pero no acaba ahí la programación, puesto que en las dos últimas celdillas de esta memoria (que suponemos de 4.096 bytes) es preciso programar el famoso «Vector de Reset». Su valor debe ser $F000 por cuanto que, como se vio en el capítulo anterior, el vector debe apuntar al comienzo del programa residente a fin de que al realizar la conexión (encendido) la CPU se dirija inmediatamente a esa dirección. Por consiguiente, introduciremos el dato $F0 en la penúltima celdilla de la memoria y $00 en la siguiente. Una vez instalada la memoria en la tarjeta estas posiciones corresponderán a las direcciones $FFFE y $FFFF.

Dicho y hecho. Ahora tenemos una memoria PROM de 4K bytes, que montamos inmediatamente en el zócalo correspondiente de nuestra tarjeta de microordenador. Conectamos la pantalla y el teclado, los cables de la alimentación y pulsamos el interruptor. Si todo se ha realizado sin errores, la pantalla estará «limpia» y podremos comenzar «a programar» en nuestro minilenguaje:

?"AB <return>

y en la pantalla aparecerá:

AB

Por el contrario, si tecleamos:

EEEEEE <return>

obtendremos, de forma implacable:

ERROR

y así sucesivamente.

Verdaderamente, hemos de admitir que no se trata de un gran logro; pero el proceso seguido para llegar a una aplicación tan microscópica y limitada puede hacerle comprender cómo es de compleja la escritura de una verdadero programa intérprete como, por ejemplo, un BASIC del Apple o del Spectrum. No obstante, en el fondo, cualquier BASIC no es otra cosa que una secuencia de instrucciones en lenguaje máquina, que obligan al hardware a esperar las órdenes procedentes del teclado y ejecutarlas si, y sólo si, no hay errores de sintaxis.

En un verdadero lenguaje «intérprete» existirán millares de pruebas y comparaciones y centenares de búsquedas de palabras clave en lugar de las dos o tres que nosotros hemos implantado, pero el resultado es siempre el mismo: la máquina, con las instrucciones proporcionadas por el software, «cobra vida» y se pone «a nuestras órdenes». Naturalmente, tendremos un «manual» bastante complejo y deberemos estudiar la sintaxis propia del lenguaje que queramos utilizar, bien sea una versión de BASIC, PASCAL, FORTH, FORTRAN o «C». Llevará bastante tiempo el aprendizaje, no pocos segundos como para nuestro microintérprete. Al final, podremos utilizar al 100% la potencia del hardware, gracias al sofisticado software que, como se dice en la jerga, «corre» dentro del ordenador.

## *Conclusiones: la importancia del software*

La ardua disertación anterior se aplica a todo ordenador, desde el más elemental de los microordenadores, hasta el más complicado «monstruo». Siempre hay un hardware, tan complejo como se quiera, pero será el software quien lo haga funcionar. Aparte de las posibilidades funcionales del hardware, únicamente las buenas cualidades del software suministrado con la máquina, o que usted desarrolle, le permitirán obtener prestaciones a alto nivel (como aprovechar al máximo la potencia del hardware). Cuanto más potente es la CPU utilizada tanto más complejo deberá ser el software, de control, normalmente denominado «sistema operativo». Instalando un software residente adecuado en un hardware este podrá realizar las funciones que se le encomienden con las máximas prestaciones, sobre todo si se le dedica a una aplicación concreta.



Este es el caso de los periféricos más importantes del ordenador personal, empezando por la impresora, y que precisamente por ello se denominan «inteligentes». Por consiguiente, las impresoras, los dispositivos de trazado («plotters»), las pantallas de texto y de gráficos, las unidades de disco y demás periféricos, son máquinas en las que existe un hardware con la habitual estructura (CPU, RAM, ROM y E/S) y un programa residente en ROM, que las hace funcionar en la forma requerida. En la práctica, los modernos periféricos son realmente ordenadores «especializados» preparados para conectarse a nuestro ordenador personal. Por ello, cuando hablemos de ellos no tendremos necesidad de describir su hardware con detalle porque, en esencia, lo conocemos ya. Por consiguiente, habrá más espacio para analizar las funciones, conexiones, manejo, virtudes y defectos. Y esto es exactamente lo que nos proponemos hacer en los capítulos siguientes.

## *GLOSARIO*

USER-FRIENDLY: Este término inglés se aplica al software que controla el funcionamiento de un ordenador haciéndolo fácil de emplear por el usuario (literalmente «amigable»). Un sistema operativo tendrá esta naturaleza cuando todas sus órdenes estén organizadas de modo que sea sencillo, para quien lo utiliza, introducirlas desde el terminal, manejar sus operaciones y aprenderlas.

FIRMWARE / SISTEMA OPERATIVO / SOFTWARE DEL SISTEMA: Todo estos términos tienen parecido significado. En general se utilizan para definir los programas que «personalizan» el hardware, confiriendo a la máquina un modo de funcionamiento ajustado a los deseos del usuario. El primer término se suele emplear para definir todo el software almacenado en memoria ROM o similar (PROM, EPROM...).

El segundo se refiere al software que controla el conjunto de las tareas efectuadas por el ordenador: asigna-

ción de E/S, de memoria, gestión de la información,... y pone a disposición del programador una serie de órdenes sencillas que le permiten acceder a zonas de este control si lo desea.

Por último, el software del sistema engloba los dos términos anteriores extendiéndose también a los lenguajes de programación.

En las máquinas modernas solamente una pequeña parte del software del sistema está en memoria ROM y con ello basta para inicializar correctamente la máquina. Así, después del encendido, son casi siempre necesarios periféricos de memoria de masa conectados al ordenador (por ejemplo, discos) de los cuales se extraen, de vez en cuando, los bloques de software necesarios para realizar las diversas operaciones. De ese modo, se ahorra mucho espacio en la memoria central del ordenador.

LENGUAJE: Programa ejecutado por la CPU del ordenador que permite al usuario escribir otros programas o introducir datos de una manera mucho más sencilla y comprensible que con el empleo del lenguaje máquina. Uno muy popular es el BASIC, que tiene la particularidad de que cuando se «arranca» —se ejecuta— es como si el usuario poseyera una nueva máquina, pues es capaz de comprender instrucciones, órdenes y programas escritos en el lenguaje BASIC. Existen muchos lenguajes para cada uno de los ordenadores presentes en el mercado, además del citado BASIC; entre ellos están los conocidos, Pascal, FORTH, FORTRAN, «C», Cobol, Logo, etc. Cada uno está recomendado para un tipo de aplicaciones, destacando en aspectos distintos.

BUCLE: Significa «anillo» o «ciclo». Es una parte integrante del programa, que se ejecuta de forma repetida hasta que se cumple una determinada condición. Un bucle, en esencia está constituido por:

1) ejecución de instrucciones, 2) pruebas para verificar si una determinada condición «impuesta antes de entrar en el bucle» es todavía válida y 3) si fuera válida, se volverá al apartado 1; si no lo fuera, terminará el bucle y la ejecución proseguirá con la instrucción inmediatamente posterior a la del apartado 3.

El empleo en un programa de un bucle, con «punteros» o «contadores» que indican el número de veces que se ejecuta el bucle, simplifica notablemente la ejecución de muchas operaciones.

SINTAXIS: Conjunto de normas que rigen un idioma y también, por extensión, un lenguaje para ordenador. Al programar, además de los errores de sintaxis, existen, naturalmente, los de ortografía. En un lenguaje, ambos errores impiden al programa intérprete comprender qué «diablo» de instrucciones le está transmitiendo el usuario, por lo que el sistema operativo se atrincherará detrás de un mensaje de error —«SYNTAX ERROR»— y corresponderá al programador la tarea de encontrar el gazapo en el programa y corregirlo.

BUG (ERROR DE PROGRAMACION): Literalmente, este término significa «chinche» y se refiere a los errores que «infactan» los programas, incluso de los más expertos programadores; se elimina con una operación adecuada de «desinfección» que toma el nombre de DEBUG (DEPURACION). Esta depuración, sobre todo en sistemas evolucionados para el desarrollo de software, se realiza por medio de un programa especial, que curiosamente, suele denominarse «DDT»; aunque lo parezca no se refiere al conocido insecticida, sino que es abreviatura de «Dynamic Debugging Tool» (literalmente herramienta de depuración dinámica) que es un rápido y potente «busca errores» utilizando, sobre todo, con programas en lenguaje máquina.

NEMONICO/ENSAMBLADOR: Cada instrucción del conjunto ejecutable por la CPU tiene su exclusivo código binario que es el único que la hace comprensible por parte del decodificador de instrucciones interno a la CPU. No obstante, sería «poco humano» obligar al programador en lenguaje máquina a recordar todos los códigos binarios (además de estúpido e inútil), por lo que el mismo fabricante de la CPU suministra una lista de las instrucciones ejecutables, asignando a cada uno de los códigos binarios un nombre fácil de recordar y que suele tner relación con lo que hace la propia instrucción. Por ejemplo: LDA por LoaD Accumulator (cargar acumulador), CLX por CLear register X (borrar registro X), STA por STore Accumulator (almacenar acumulador), STAX por STore Accumulador indexed X (almacenar acumulador indexado X), etc,

El programador escribe su programa empleando estos códigos nemónicos, de forma que las instrucciones coinciden, una a una, con las ejecutables por la CPU. El lenguaje utilizado se denomina lenguaje Ensamblador. Una vez escrito el programa en lenguaje Ensamblador («programa fuente»), se debería consultar el manual de la CPU y traducir cada código nemónico al código binario correspondiente. Afortunadamente, esta operación se realiza de forma automática en muchas máquinas, con programas de traducción denominados «ensambladores», que realizan justamente esa traducción, de forma que al final se tiene unas secuencia de códigos binarios que representan el programa en formato ejecutable por la CPU («programa objeto»). Para que se puede finalmente ejecutar debe introducirse dicho programa en memoria: sea RAM o ROM (EPROM,...) y posicionar el contador de programa en la dirección de comienzo.



# CAPITULO V

## SISTEMAS DE NUMERACION Y CODIFICACION Y ALGUNOS TRUCOS ARITMETICOS

EN los capítulos anteriores hemos manejado con desparpajo números binarios y hexadecimales, pero hay todavía muchas cosas que precisar sobre ellos: cómo se representa un número negativo, cómo se opera con ellos, de qué forma pasar de uno a otro...

También debemos tratar los sistemas de codificación, de los que no hemos hablado nada hasta ahora y que, sin embargo, tienen bastante interés.

### *Números binarios*

Comencemos por el principio: en un ordenador todas las señales son eléctricas; es decir, del tipo hay o no tensión. ¿En dónde? Naturalmente, en un hilo que une dos dispositivos, de los cuales uno envía y el otro recibe la señal. Llamamos "1" al estado de "tensión en el hilo" y "0" al estado contrario, según un convenio universalmente admitido. Como hemos visto, en muchos casos se reunen hilos que tienen la misma función, hasta formar un haz al que están conectados la CPU y los diversos dispositivos, que de esta forma pueden dialogar a través de esta vía común que se denomina bus.

Así, podemos definir simultáneamente el estado de se-

ñal en todos los hilos o líneas del bus definiendo un dato, expresado con "ceros" y "unos, formado por tantos dígitos como hilos o líneas haya en el bus. Para distinguir entre sí estas líneas, se atribuye a cada una de ellas un nombre y una prioridad: a la primera línea corresponderá el llamado "bit número 0", a la segunda el "bit número 1" y así sucesivamente. La atribución de la prioridad se resuelve confiriendo mayor "peso" (importancia) a los bits cuya denominación sea más "alta": el bit 0 será el de menor peso, o lo que es lo mismo el bit "menos significativo" (LSB = Least Significant Bit), mientras que el bit 7 (sobre 8) o el bit 15 (sobre 16) son los bits "más significativos" (MSB = Most Significant Bit).

De esto se deduce que para un bus de 8 líneas, por ejemplo el bus de datos, el número resultante puede ser una combinación cualquiera de "unos" y de "ceros" desde 0000.0000 hasta 1111.1111, mientras que para un bus de 16 líneas (por ejemplo el bus de direcciones) las combinaciones son todas las comprendidas entre 0000.0000.0000.0000 y la 1111.1111.1111.1111 (hemos añadido puntos subdividiendo los números en grupos de cuatro dígitos, simplemente por comodidad de lectura). En el primer caso tendremos solamente 256 combinaciones posibles, mientras que en el segundo caso las combinaciones serán 65.536. Todos estos números se denominan "binarios", ya que cada dígito sólo puede ser 0 ó 1. El dígito simple toma el nombre de "bit" (unidad elemental de información), contracción de la denominación inglesa "Binary digit" (dígito binario). Ahora, finalmente, estamos preparados para traducir los números binarios a una numeración que todos debemos conocer: la numeración decimal, es decir, realizar conversiones entre números expresados en base 2 ó 10.

Para hacerlo, basta construir una tabla de correspondencia como la siguiente:

| número binario | número decimal correspondiente |
|---|---|
| 0000.0000 | 0 |
| 0000.0001 | 1 |
| 0000.0010 | 2 |
| 0000.0011 | 3 |
| ... | ... |
| 1111.1111 | 255 |
| ... | ... |



y así sucesivamente. Se podría continuar hasta el "infinito", aumentando el número de bits, pero un procedimiento de este tipo es inútil. Lo más práctico (a no ser que usted "invente" otra cosa) es efectuar en cada ocasión, el cálculo siguiente:

Valor decimal = bit 0 × $2^0$ + bit 1 × $2^1$ + bit 2 × $2^2$ + ... + bit n × $2^n$

en donde «$2^n$» significa «2 elevado a la enésima potencia»; el bit 0 es el valor del bit menos significativo (LSB) y el bit n es el del bit más significativo (MSB). Por ejemplo, el número binario 0101.0000 corresponde al valor decimal 80, ya que se tiene: $0 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 0 \cdot 2^3 + 1 \cdot 2^4 + 0 \cdot 2^5 + 1 \cdot 2^6 + 0 \cdot 2^7 = 16 + 64 = 80$.

De ahora en adelante, y cuando se puedan prestar a confusión usaremos la siguiente notación para determinar en qué base está el número que escribamos:

| | Decimal | Hexadecimal | Binario |
|---|---|---|---|
| Expr. equivalentes | 10 | $0A | %00001010 |
| | 10d | 0Ah | 00001010b |

## *Sistema de numeración hexadecimal*

La numeración binaria, tan cómoda para el ordenador, es notablemente incómoda para los usuarios. Esta es larazón del uso de los números hexadecimales.

La numeración hexadecimal no es más que un «refrito» de la numeración binaria, mediante el cual podemos tratar números equivalentes a los binarios sin tener que manejar grandes cantidades de «ceros» y de «unos».

El truco (que, en definitiva, se basa en el hecho de que 16 —base de la numeración hexadecimal— es la cuarta potencia de 2 —base de la binaria—) es el siguiente: se agrupan los «ceros» y los «unos» del número binario de cuatro en cuatro bits partiendo, por su puesto, del bit menos significativo. Habida cuenta de que con cuatro bits se obtienen hasta 16 combinaciones, podemos utilizar 16 símbolos para representar sin confusión todas estas combinaciones. De este modo, habremos «comprimido» 4 dígitos binarios (bits)

en un solo dígito hexadecimal. Los 16 símbolos elegidos fueron las 10 cifras decimales (0,1,...,9) más las 6 primeras letras del alfabeto A, B, C, D, E, F. De este modo cada grupo de 4 bits (denominado también «nibble») se corresponde con un dígito hexadecimal, como se puede ver en la siguiente tabla:

| DECIMAL | HEXADECIMAL | BINARIO |
|---|---|---|
| 0 | 0 | 0000 |
| 1 | 1 | 0001 |
| 2 | 2 | 0010 |
| 3 | 3 | 0011 |
| 4 | 4 | 0100 |
| 5 | 5 | 0101 |
| 6 | 6 | 0110 |
| 7 | 7 | 0111 |
| 8 | 8 | 1000 |
| 9 | 9 | 1001 |
| 10 | A | 1010 |
| 11 | B | 1011 |
| 12 | C | 1100 |
| 13 | D | 1101 |
| 14 | E | 1110 |
| 15 | F | 1111 |

Con la notación hexadecimal una dirección como 111.111.111.1110b adopta la forma más humana y comprensible de FFFEh (o bien $FFFE).

Con 8 bits los números hexadecimales correspondientes tendrán dos dígitos,desde 00h a FFh; con 16 bits tendrán cuatro dígitos, desde 0000h a FFFh, etc. Mediante el empleo de los números hexadecimales resulta también más fácil encontrar el valor decimal correspondiente, ya que cada cifra hexadecimal representa a cuatro bits (los pesos son potencias de 16 y no de 2) el valor decimal se calculará aplicando la fórmula siguiente:

valor decimal = dígito 0 × $16^0$ + dígito 1 × $16^1$ + ... + dígito n × $16^n$.

Si el número es, por ejemplo, 50h, el valor es 0 × 1 + 5 × 16 = 80 (Recordemos que la potencia «cero» de cualquier número es 1; $16^0 = 1$). Es conveniente habituarse, desde el principio, a manejar los números en sus distintas representaciones (decimal, binaria, hexadecimal o, como veremos más adelante, en BCD), pues el buen programador deberá usarlas a menudo.

Quizás crea saber cómo se leen los números. Tanto si es así como si modesta y atinadamente, cree que no, veámoslo:

- Un número decimal se lee como estamos acostumbrados: <uno>, <diez>, <cien>..., etc.
- Un número binario se lee dígito a dígito; así, 1001.0111b, por ejemplo, se lee «uno.cero.cero.uno.cero.uno.uno.uno», y 101b «uno.cero.uno» y no «ciento uno».
- Un número hexadecimal se lee también dígito a dígito, incluso si se parece a un número decimal. Por consiguiente, AF3Ch se lle «a.efe.tres.ce» y, por ejemplo, «1000» se lee «uno.cero.cero.cero» y no «mil» como hace algún despistado.

### *Un poco de aritmética binaria*

También los números binarios o hexadecimales, como todos los números que se precien de serlo, se pueden sumar, restar, multiplicar, dividir. Las reglas son idénticas a las que se aplican a la aritmética decimal, con la única diferencia de los «pesos», que son potencias de 10 en el sistema decimal, de 2 en el binario y de 16 en el hexadecimal. Teniendo en cuenta los acarreos («carry»), es decir, el exceso sobre el mayor dígito de la base, las cuentas siempre saldrán exactas. Por ejemplo 12d + 19d = 31d tiene un acarreo desde las unidades a las decenas. Por el contrario, en binario se tendría para la misma operación 0000.1100b + 0001.0011b = 10001.1111b que no produce ningún acarrero; también en hexadecimal se tendría OCh + 13F= 1Fh, que no genera acarreos desde un peso a otro.

En binario se produce un acarreo cuando se suman dos dígitos iguales a uno, dado que 1b + 1b = 10b. Así, por ejemplo: 7d + 12d = 19d (sin acarreo); pero su equivalente en binario 0000.0111b + 0000.1100b = 0001.0011b o también 7h +

Ch = 13h, tienen un acarreo desde el grupo de 4 bytes («nibble») menos significativo al inmediato superior.

Para practicar puede utilizar la rueda de la Figura 1. Cada vez que se pasa por el cero se tiene un acarreo; de forma análoga se pueden calcular las sustracciones, teniendo en

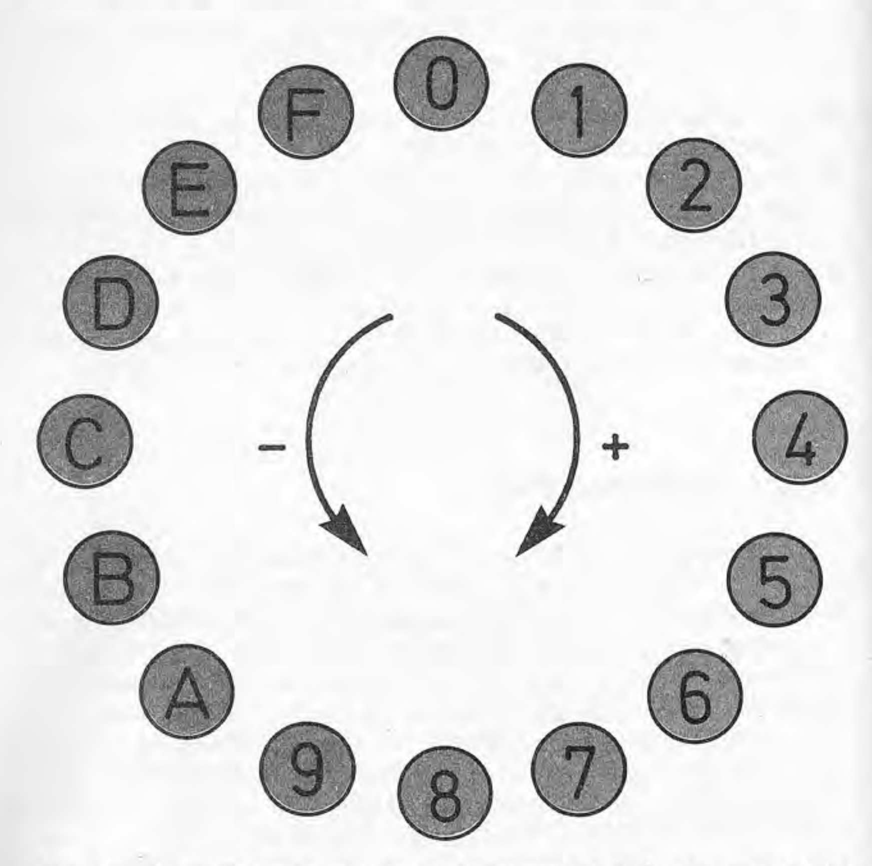



*Fig. 1 - Rueda para el cálculo de "acarreos" en las sumas o restas de dígitos hexadecimales. Para una suma, se parte del dígito del primer término y se gira en sentido horario tantas posiciones como indique el dígito del segundo término, con un acarreo de 1 cada vez que se pasa por el cero: por ejemplo, 1Ch + 9h = 5h con un acarreo de 1 a sumar al bit de mayor peso, es decir, en total 25 h. Para una resta se gira en sentido contrario, con un acarreo negativo de 1 cada vez que se pasa por cero: 25h 9h se calcula tomando 5h, retrocediendo en 9 posiciones hasta Ch y contando -1 de acarreo negativo a quitar del dígito de peso superior. El resultado definitivo es 1Ch.*



cuenta que, en este caso, el acarreo es negativo (en inglés «borrow», que vale -1).

## *Números negativos*

Hemos visto que todos los números decimales pueden representarse, de forma biunívoca, en binario (o en hexadecimal); para ello bastará disponer del adecuado número de bits. Con 8 bits se pueden representar todos los números decimales desde 0d a 255d, mientras que con 16 bits se tienen todos los valores comprendidos entre 0d y 65535d; como observará todos los valores mencionados son positivos. Sin embargo, a menudo tenemos que trabajar con valores negativos. En tal caso, el artificio más difundido consiste en utilizar el bit de mayor peso (el bit 7 en 8 bits y el 15 en 16) —recordemos que el menos significativo es el bit 0— para indicar el signo, de modo que si es cero el número es positivo y si es 1 el número es negativo. Utilizando este convenio, con 8 bits podemos representar los números comprendidos entre -128d y +127d, y con 16 bits se pueden representar los números comprendidos entre -32768d y +32767d.

Por comodidad, el tratamiento de los números enteros negativos se realiza representándolos mediante una técnica especial denominada «en complemento a 2». A primera vista, se podrá pensar que, si 000.0001b es el valor 1d, el valor -1d estaría representado por 1000.0001b. En cambio, se utiliza un método menos evidente, pero más productivo, que consiste en que el número negativo se obtiene tomando el número positivo correspondiente, sustituyendo los «ceros» por «unos», y viceversa, y sumándole luego al resultado el valor 1 (teniendo en cuenta, claro está, los posibles acarreos a los pesos superiores). Por consiguiente, «-1d» se hace 0000.0001b(+1d) ⟶ 1111.1110b (complemento a 1) ⟶ 111.1111b(-1d). Este sistema no es demasiado complejo y, en compensación, facilita mucho la ejecución de cualquier operación en la CPU.

Para simplificar la circuitería de la unidad aritmético-lógica (ALU), dentro de la unidad central de proceso (CPU) no existe una lógica de sustracción, sino solamente un bloque sumador. Cuando se tienen que sumar dos números bi-

narios no hay problema alguno, pero cuando se tiene que efectuar una resta, el sustraendo se pone antes automáticamente en complemento a dos de forma que la resta se transforma en una simple suma. A modo de prueba veamos un ejemplo: 1d - 1d = 0d (sobre este resultado no hay duda alguna) se convierte en 0000.0001b (1d) + 1111.1111b (1-d) = 000.000b como se quería demostrar. El acarreo desde el bit más significativo no se tiene en cuenta puesto que el segundo término de la suma es un número negativo y la CPU «lo sabe»).

Si quiere convencerse todavía más, puede consultar cualquier libro de la bibliografía. Encontrará, en la descripción de los circuitos digitales todos los sumadores y «negadores» que quiera, pero ningún «restador». En resumen: cualquier suma en binario (o en hexadecimal) se calcula de modo análogo a como se efectúa con los números decimales, teniendo en cuenta los acarreos de un peso a otro; por el contrario, toda resta se efectúa sustituyendo el sustraendo por su complemento a 2, y sumándolo luego al minuendo en lugar de restarlo. Esta operación es realizada de forma automática por la ALU de la CPU, o bien deberemos prepararla (véase Tabla 1), dependiendo del $\mu$P que usemos.

Números negativos ⟵ 2.° dígito hexad.

| 1.$^{er}$ dígito hex. ↓ | | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| | E | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| | D | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 44 |
| | C | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 |
| | B | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 |
| | A | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 |
| | 9 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 |
| | 8 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 | 128 |

*Tabla 1 - Tabla para el cálculo de los números negativos de 8 bits (dos dígitos hexadecimales). Por comodidad, no ponemos el signo "-". Para obtener, por ejemplo, -20 se busca 20 (por sencillez se obvia el signo menos) y se obtiene EC. Se puede verificar el resultado en el modo habitual: 20d = 0001.0100b; el complemento a 1 es 1110.1011b y el complemento a 2 (se suma 1) es 1110.1100b = ECh, como se esperaba.*



## Conversión de un número decimal a su equivalente binario (o hexadecimal)

En los apartados anteriores se constató cómo cualquier número binario, o hexadecimal, es fácilmente convertible en su valor decimal. Un poco más compleja resulta la operación inversa, aunque siempre es posible. Basta tomar el valor decimal que se quiere convertir a binario, dividirlo por dos y apuntar el resto de la división, que será naturalmente 1 ó 0. El cociente se divide nuevamente por dos, marcando también el resto y así se continúa hasta que el cociente sea igual a 0. El último resto, será el bit más significativo y los restos anteriores son, por orden, los bits de pesos decrecientes.

En la Figura 2 se ilustra este procedimiento con dos ejemplos sencillos que se leerán como sigue: a) «251d dividido por 2 es 125d con resto 1; 125d dividido por 2 es 62d con resto 1; 62 d dividido por 2 es 31d con resto 0; 31d dividido por 2 es 15d con resto 1; 15d dividido por 2 es 7d con resto 1; 7d dividido por 2 es 3d con resto 1; 3d dividido

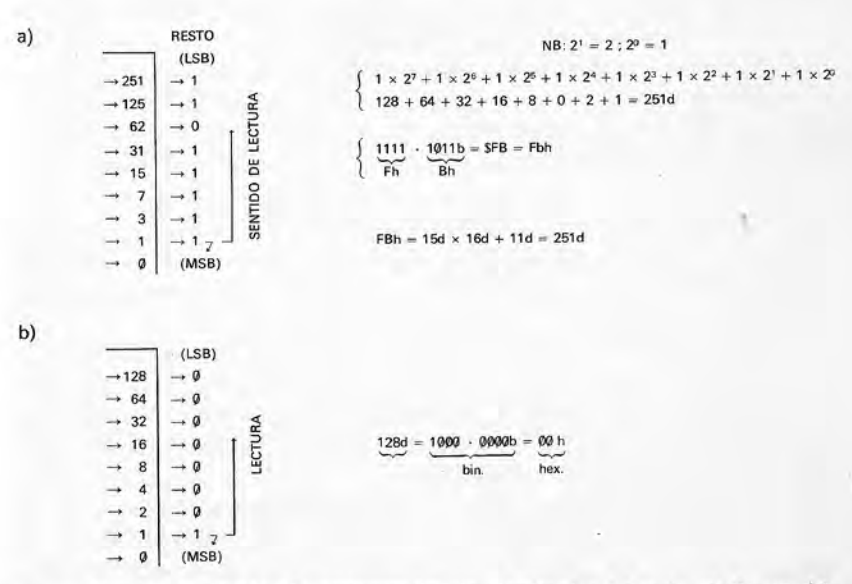



*Fig. 2 - Ejemplo de conversión de decimal a binario a) de un número impar, b) de un número par.*

por 2 es 1d con resto 1; 1d entre 2 es 0 con resto —último resto— 1». El número binario resulta, pues: 111.1011b = FBh = 251d como se puede comprobar con facilidad. Un ejemplo análogo es el de la Figura 2b, en donde el número a convertir es par. Para convertir un número negativo se transforma previamente sin tener en cuenta el signo y luego se realiza su complemento a dos, tal como se vio en el apartado anterior.

## Código BCD: decimal codificado en binario

Hay ocasiones en las cuales en lugar de convertir, por eemplo, el número «9781d», aplicando el procedimiento antes indicado, nos conviene convertir directamente cada dígito decimal en su equivalente binario (ya que con 4 bits podemos representar todos los dígitos decimales desde 0 a 9); un caso típico de esta necesidad es la presentación en un display de 7 segmentos (7 leds colocados como un 8: 8) de un número decimal cualquiera.

Para nuestro 9781d obtendremos el número binario 1001.0111.1000.0001 que, evidentemente no es el número binario correspondiente. De hecho, al convertir este número a decimal aplicando las fórmulas que vimos anteriormente se tendría 39025d, bien diferente del 9871d. Todo ello sucede porque hemos convertido el número decimal de partida utilizando la notación «decimal codificado en binario» (Binary Coded Decimal = BCD). Se trata de una notación muy cómoda para el usuario que, aunque estemos en la era del ordenador, sigue utilizando diez dedos y, por consiguiente, el sistema decimal.

Realmente los ordenadores, para muchas aplicaciones de entrada/salida dirigidas a nosotros e incluso en el cálculo científico o comercial, se adaptan a nuestras exigencias y adoptan el sistema BCD (para las cantidades muy grandes o muy pepeñas se utilizará la denominada representación en «coma flotante», pero este tema se tratará al ver el BASIC). Es evidente que con este sistema despilfarramos bits respecto al sistema binario puro (un 20% como mínimo). Por ejemplo, 999d requiere tres dígitos BCD (1001.1001.1001) es decir, 12 bits, frente a los 10 bits necesarios en el sistema binario puro o hexadecimal. Al efectuar operaciones con dígitos en BCD, hemos de tener presente que ya no se producen acarreos al pasar por el 0 después del valor F(1111b), si no después del valor 9d(1001 bcd). Una buena ALU permite al menos efectuar sumas en BCD.



## La codificación ASCII

En un ordenador las configuraciones de «ceros» y «unos» dejan muy pocas oportunidades a la fantasía del programador. No solamente los bytes tienen un significado bien preciso (sea en binario/hexadecimal o en BCD), sino que también, cuando queramos representar caracteres alfanuméricos, estaremos ligados a codificaciones estándar universalmente utilizadas. Una de ellas, la más conocida,es la codificación ASCII, abreviatura de «American Standar code for Information Interchange» y que utiliza 7 bits para definir 128 códigos diferentes (el octavo bit es siempre 0 para los caracteres estándar, mientras que vale 1 cuando se refiere a caracteres semigráficos, particulares de algunas máquinas). Cada uno de estos 128 códigos corresponde a una letra del alfabeto, a un número de 0 a 9, a un signo de puntuación, a un código de control, etc., tal como se pone de manifiesto en la Tabla 2, donde para cada carácter se indica su codificación ASCII expresada en decimal y hexadecimal.

Cuando enviemos, por ejemplo, el código ASCII 41h, estaremos representando la letra «A». ¿En dónde utilizamos los códigos ASCII para codificar letras, números y símbolos diversos? Cuando un «port» de entrada «lee» el teclado de nuestro ordenador personal, por ejemplo, al pulsar la tecla «A» un circuito particular acoplado al teclado (conocido como «codificador»), sitúa el código ASCII 0100.0001b es decir, 41h, en las 7 líneas que están conectadas a dicho «port» de entrada. Desde ese momento, dentro del ordenador, el dato 41 h indicará la letra «A», y todos los dispositivos capaces de comprender la codificación ASCII lo considerarán automáticamente como tal. Por ejemplo, un controlador de video al que la CPU envía 41h visualizará en la pantalla «A», una impresora imprimirá «A», etc.

Con 7 bits, y por consiguiente con un conjunto de 128 caracteres diferentes, se pueden «traducir» todos los símbo-

## CODIGOS ASCII NORMALIZADOS

| Dec. | Hex. | CAR. | Dec. | Hex. | CAR. | Dec. | Hex. | CAR. |
|---|---|---|---|---|---|---|---|---|
| 000 | 00 | NUL | 043 | 2B | + | 086 | 56 | V |
| 001 | 01 | SOH | 044 | 2C | , | 087 | 57 | W |
| 002 | 02 | STX | 045 | 2D | - | 088 | 58 | X |
| 003 | 03 | ETX | 046 | 2E | . | 089 | 59 | Y |
| 004 | 04 | EOT | 047 | 2F | / | 090 | 5A | Z |
| 005 | 05 | ENQ | 048 | 30 | 0 | 081 | 5B | [ |
| 006 | 06 | ACK | 049 | 31 | 1 | 092 | 5C | \ |
| 007 | 07 | BEL | 050 | 32 | 2 | 093 | 5D | ] |
| 008 | 08 | BS | 051 | 33 | 3 | 094 | 5E | ^ |
| 009 | 09 | HT | 052 | 34 | 4 | 095 | 5F | — |
| 010 | 0A | LF | 053 | 35 | 5 | 096 | 60 | ` |
| 011 | 0B | VT | 054 | 36 | 6 | 097 | 61 | a |
| 012 | 0C | FF | 055 | 37 | 7 | 098 | 62 | b |
| 013 | OD | CR | 056 | 38 | 8 | 099 | 63 | c |
| 014 | 0E | SO | 057 | 39 | 9 | 100 | 64 | d |
| 015 | 0F | SI | 058 | 3A | : | 101 | 65 | e |
| 016 | 10 | DLE | 059 | 3B | ; | 102 | 66 | f |
| 017 | 11 | DC1 | 060 | 3C | < | 103 | 67 | g |
| 018 | 12 | DC2 | 061 | 3D | = | 104 | 68 | h |
| 019 | 13 | DC3 | 062 | 3E | > | 105 | 69 | i |
| 020 | 14 | DC4 | 063 | 3F | ? | 106 | 6A | j |
| 021 | 15 | NAK | 064 | 40 | | 107 | 6B | k |
| 022 | 16 | SYN | 065 | 41 | A | 108 | 6C | l |
| 023 | 17 | ETB | 066 | 42 | B | 109 | 6D | m |
| 024 | 18 | CAN | 067 | 43 | C | 110 | 6E | n |
| 025 | 19 | EM | 068 | 44 | D | 111 | 6F | o |
| 026 | 1A | SUB | 069 | 45 | E | 112 | 70 | p |
| 027 | 1B | ESCAPE | 070 | 46 | F | 113 | 71 | q |
| 028 | 1C | FS | 071 | 47 | G | 114 | 72 | r |
| 029 | 1D | GS | 072 | 48 | H | 115 | 73 | s |
| 030 | 1E | RS | 073 | 49 | I | 116 | 74 | t |
| 031 | 1F | US | 074 | 4A | J | 117 | 75 | u |
| 032 | 20 | SPACE | 075 | 4B | K | 118 | 76 | v |
| 033 | 21 | ! | 076 | 4C | L | 119 | 77 | w |
| 034 | 22 | " | 077 | 4D | M | 120 | 78 | x |
| 035 | 23 | # | 078 | 4E | N | 121 | 79 | y |
| 036 | 24 | $ | 079 | 4F | O | 122 | 7A | z |
| 037 | 25 | % | 080 | 50 | P | 123 | 7B | { |
| 038 | 26 | & | 081 | 51 | Q | 124 | 7C | \| |
| 039 | 27 | ' | 082 | 52 | R | 125 | 7D | } |
| 040 | 28 | ( | 083 | 53 | S | 126 | 7E | ~ |
| 041 | 29 | ) | 084 | 54 | T | 127 | 7F | DEL |
| 042 | 2A | * | 085 | 55 | U | | | |

Dec = decimal, Hex = hexadecimal, CAR = carácter
LF = Line Feed (avance línea), FF = Form Feed (avance una hoja papel),
CR = Carriage Return (retorno de carro), DEL = borrado

*Tabla 2 - Tabla de los 128 códigos ASCII normalizados (bit 7 = 0), incluyendo los códigos de control.*



los alfanuméricos junto con muchos otros caracteres especiales: <!,.?."#$&'( )*;=-> y diversas señales de control.

Se trata de códigos importantísimos que sirven para el intercambio de información incluso bajo la forma de códigos de control que, como se ve en la Tabla 2, corresponden a los que van desde 1d a 31d y al 127; incluyendo señales de control tales como BEL (timbre), CR (Carriage Return-retorno de carro), BS («Back Space», retrocede un espacio), DEL (borrado) y otros controles más «comunicativos» tales como ACK («Acknowledgement», reconocimiento), NACK (Non-Acknowledgement» —no reconomiento), etc.

Los primeros sirven para transferir al periférico, que suele ser una impresora (pero que también puede ser un terminal de video o dispositivos similares), la orden de hacer sonar alguna señal acústica (si la hubiere), del retorno del carro de impresión, o bien generar un salto de línea (LF, Line Feed) o un salto de página (FF, Form Feed). Por el contrario, los códigos de control tales como ACK y NACK sirven para posibilitar el intercambio de información entre los periféricos y el ordenador, con señales de tipo «recibido» o «no recibido», la clásica de «corto y cierro», etc.

Los caracteres de control se generan en el teclado de un ordenador personal pulsando simultáneamente una tecla denominada de control (abreviada como CTRL con frecuencia) y de la una letra. Las letras utilizadas van desde la A en adelante. Por ejemplo, Control (lo abreviamos con el signo ^) A proporciona el código ASCII 001d, que significa «Start of Header» Comienzo de cabecera (SOH), ^F(006d) da ACK, ^J(010d) proporciona LF, ^M(013d) da CR, etc.

Si, en BASIC, se teclea:

```
PRINT CHR$ (7)
```

que significa «escribe en la pantalla el valor del código ASCII 7», en realidad no se escribe nada si no que el ordenador emite un tono de corta duración («beep»), que corresponde al carácter BEL que se indica en la Tabla 2.

El código 027d (Escape) indica que la secuencia de caracteres que va después del mismo debe considerarse de un modo particular (se aparta del estándar) y permite a los dos interlocutores entenderse según convenios particulares.

No hay ninguna razón para usar otra codificación que

no sea la ASCII, pues todos los dispositivos, y sobre todo los periféricos de uso más frecuente, están diseñados de modo que admitan sus códigos: comprenderán perfectamente los datos que se les transmitan siempre que estén expresados, precisamente, en los correspondientes códigos ASCII.

En nuestro ejemplo del capítulo anterior, por el contrario, utilizamos una codificación «espartana» e inmediata para las teclas de entrada, lo cual no importa demasiado porque suponíamos que se disponía de todo un hardware «hecho a medida» y el ejemplo era meramente didáctico. En realidad, si hubiéramos usado un verdadero teclado comercial, y un monitor de video ordinario, las teclas E, R, O, A, B, " y ? se hubieran codificado, en ASCII, como 45h, 52h 4Fh, 41h, 42h, 22h y 3Fh, correspondientes a los valoresdecimales 69d, 82d, 79d, 65d, 66d, 34d y 63d.

Quienquiera que tenga un ordenador personal con una versión del lenguaje BASIC, si no se fía, puede comprobar dichas codificaciones internas probando la instrucción PRINT ASC («x»), poniendo en lugar de <<x>>, sucesivamente (lo que constituye un ejercicio banal, pero instructivo) todos los caracteres del teclado.

Y ya va siendo hora de hablar en profundidad de uno de los periféricos «por excelencia» de un sistema informático: la impresora.



# CAPITULO VI

## LA IMPRESORA

POR fin podemos hablar de los periféricos de un ordenador es decir, de todas aquellas máquinas y/o accesorios que se conectan a la estructura base de nuestra unidad de proceso. El mercado ofrece numerosísimos tipos, que desarrollan las funciones más variadas y especializadas, pero nos limitaremos a darles una descripción resumida, de la mayoría concentrando nuestra atención en los más importantes. En este capítulo hablaremos de las impresoras que, después de la pantalla y del teclado (que constituyen un conjunto de E/S considerado normalmente parte integrante del ordenador personal) es el periférico «número uno» de nuestro sistema. Su importantísima función es plasmar de manera permanente sobre papel el fruto de los procesos de nuestro ordenador, tales como el listado de un programa, sus resultados numéricos o gráficos, o un conjunto de datos extraído de cualquier dispositivo de almacenamiento (sea o no de memoria masiva).

La importancia de tener una impresión de los datos es evidente: en papel se puede razonar mejor, se puede tener una visión de conjunto de los datos, como si se leyera un libro, y se pueden efectuar también más fácilmente correcciones teniendo siempre a la vista todas las variaciones aportadas con respecto al original sacado por la impresora. Todas estas operaciones son imposibles, o por lo menos bas-



*Foto 1 - Moderno teletipo.*

tante difíciles de efectuar, con la ayuda de una pantalla, que solamente permite la visión de una parte limitada del programa o de una página cada vez, por no hablar de la incomodidad que supone tener que volver a encender la máquina y reinsertar los disquetes, después de apagarla, si queremos ver de nuevo los datos. En relación con la pantalla, pues, las ventajas son notables mientras que los inconvenientes son exclusivamente la relativa lentitud de la impresión y el consumo de papel.

De todas formas, precisamente por eso, la impresora no puede sustituir por completo a la pantalla, pues a esta última corresponde desempeñar las funciones simples y rápidas de edición de los datos introducidos por el teclado, por lo que son válidas las reglas de oro siguientes: 1) utilizar la pantalla para trabajar durante la introducción de los datos; 2) emplear la impresora una vez finalizada la introducción para detectar con tranquilidad, sentados en la mesa, los errores que hayamos cometido o bien tener un listado por escrito.

No obstante, insistimos en que, por regla general, cualquier ordenador personal que se precie tendrá un teclado y una pantalla, bien integrados en la estructura de base o





*Foto 2 - Una de las últimas impresoras de agujas: la MicroL 93A de OKI.*

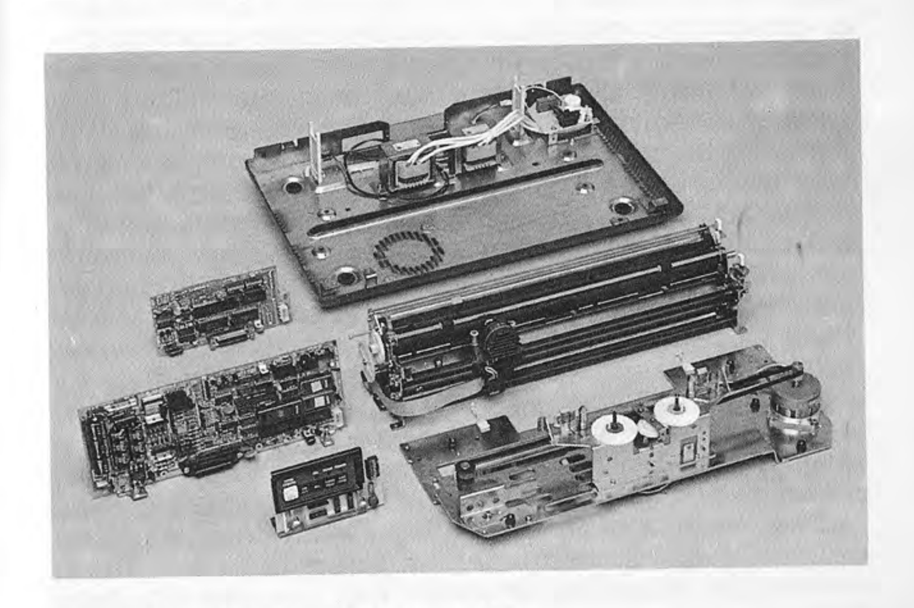


*Foto 3 - Despiece de la MicroL 93A.*

externos (las funciones y características generales de las pantallas serán examinadas más adelante).

### *Impresoras sin impacto*

En esta categoría se incluyen todas aquellas impresoras que imprimen los caracteres en el papel sin ayuda de medios mecánicos de percusión, a diferencia de lo que sucede, por ejemplo, en una máquina de escribir normal, en la cual es una especie de «martillo» el que marca el símbolo sobre el papel, al golpear una cinta entintada. Las tecnologías de «no percusión» son numerosas y una de sus mejores cualidades es el bajo nivel de ruido que producen, lo que se debe a que durante la impresión hay muy pocas partes en movimiento; por ello, son siempre aceptadas en todos aquellos ámbitos de trabajo en donde el silencio es oro.

Comenzaremos por los modelos más económicos: las impresoras térmicas. Suelen basarse en mecanismos muy simples, adecuados para máquinas de dimensiones reducidas, como lo prueba su empleo cada vez más frecuente en los ordenadores personales portátiles. Estas impresoras se utilizan mucho también en dispositivos basados en microprocesadores y dedicados a funciones específicas, tales como cajas registradoras, balanzas y registradores de datos («data-logger»). Un denominar común de las impresoras térmicas es la reducida longitud de la línea de impresión, que puede contener un máximo de 80 caracteres en los modelos más sofisticados, mientras que la velocidad de impresión varía de un modelo a otro y es inversamente proporcional a la longitud de la línea. Por lo general, son velocidades bastante frecuentes de 2 a 5 líneas por segundo para los modelos de 20 caracteres por línea.

El papel sobre el que se imprimen los caracteres está tratado por medios químicos con una sustancia sensible al calor (papel termosensible). Este se arrasta a velocidad constante o, dicho más propiamente, con pequeños saltos hacia adelante, uniformemente separados. Mientras se desplaza, un rodillo lo oprime contra un soporte, denominado «cabeza de impresión», en donde hay una serie diminutos elementos que se pueden o no calentar. Cada uno de ellos está controlado por una de las líneas procedentes de un



«port» de salida que, a su vez, está conectado al resto del hardware del ordenador. En este último debe existir un programa de control que envíe todos los impulsos de forma correcta al «port» y, por consiguiente, a los elementos de impresión. Cuando se activa un elemento, la corriente que lo atraviesa lo calienta, ya que en la práctica dicho elemento no es otra cosa que una resistencia muy pequeña. El calor generado, aunque el tiempo de activación sea bastante breve, es suficiente como para hacer oscurecer el papel en el punto que, en ese preciso instante, está en contacto con el elemento de impresión, y así en el papel quedará marcado un minúsculo punto. Con el desplazamiento de la cabeza perpendicularmente al sentido de arrastre del papel, se consigue «incidir» en una serie ordenada de puntos y, por consiguiente, crear una serie de caracteres cuya definición depende exclusivamente de las dimensiones del punto generado y de la proximidad de los elementos de caldeo entre sí.

Dentro de ciertos límites, cuantos más sean los elementos de caldeo de la cabeza tanto más alta puede ser la velocidad de impresión, ya que nada impide activar simultáneamente más elementos para formar en el papel más caracteres en una sola pasada.

Un segundo tipo de impresora de «no percusión» es la que utiliza un papel que no es sensible al calor, sino a las descargas de electricidad estática. En tal caso, se habla detecnología de impresión electrostática. El papel tiene una superficie plateada y se ennegrece donde se produce la descarga eléctrica, es decir, entre el elemento de la cabeza y el propio papel. Por lo demás, la mecánica y la técnica de impresión (movimiento de la cabeza y arrastre del papel) son las mismas que los de la impresora térmica. En la Figura 1 se ilustra, de forma simplificada, la mecánica de estos dos tipos de impresora.

Un tercer tipo de impresora tiene una cabezas móvil, que se desplaza perpendicularmente al sentido de arrastre del papel que, en la práctica es un embudo muy delgado por el que, desde un depósito especial, se hacen pasar microgotas de tinta que luego «embadurnan», naturalmente de forma controlada, el papel situado delante. Una impresora que utiliza esta tecnología de impresión toma el nombre de «Ink-jet» (es decir, de chorro de tinta).

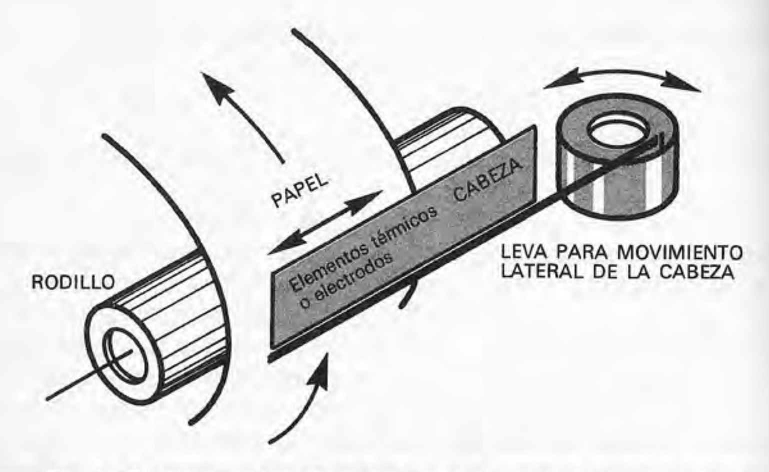



*Fig. 1 - Estructura simplificada de una impresora térmica o electrostática.*

Es evidente que, aparte del método empleado, el objeto es siempre el mismo: generar los caracteres en el papel componiéndolos por medio de puntos.

En cualquier caso, el arrastre del papel y de la cabeza en sentido perpendicular, son controlados siempre por un microprocesador (naturalmente, con su RAM, ROM y E/S). Finalmente, uno de los más recientes métodos de impresión es el «disparo de tinta», tecnología utilizada en algunas máquinas producidas por Olivetti. Este método (en la terminología inglesa denominado «Ink-shot») combina la modalidad de impresión electrostática con la de chorro de tinta, eliminando los problemas comunes a ambas.

El papel electrostático y el termosensible se alteran, sobre todo si se exponen a la acción de la luz, y además las técnicas de tipo «ink-jet» producen frecuentes problemas en la conservación de la cabeza de impresión. Por el contrario, con el sistema «ink-shot» se utiliza papel normal, por consiguiente no sujeto a deterioro y la tinta, sólida, está contenida en un cartucho especial alojado en la cabeza móvil. El papel se desplaza entre la punta del cartucho (ánodo) y una placa de soporte (cátodo). Cuando llega el momento de generar un punto del carácter objeto de impresión se produce una descarga entre los dos electrodos y una parte de la tinta se disgrega, a causa de la microexplosión, siendo arras-



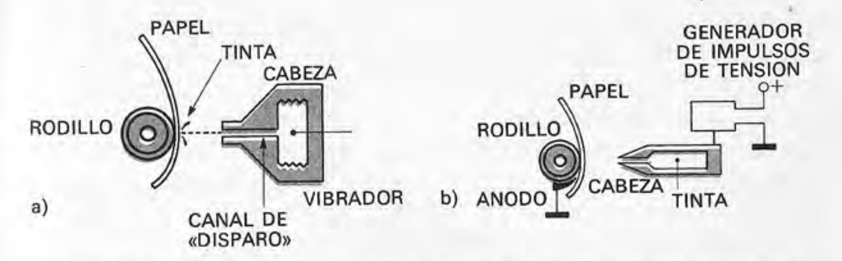



*Fig. 2 - Estructura simplificada de una impresora de chorro de tinta ("ink-jet") (a) y de disparo de tinta ("ink-shot") (b).*

trada hasta el papel en forma de microgotas que, al golpear, forman el habitual punto. En la Figura 2 se ilustran ambos tipos de impresión.

Recursos similares a este (por ejemplo, la técnica «Think jet» de Hewlett Packard, que proyecta las gotas por calentamiento de la cabeza) están haciendo cada vez más fiables y ventajosas las tecnologías de impresión por chorro de tinta.

No queremos concluir el tema de las impresoras de «no percusión», sin citar las nuevas y avanzadas tecnologías que hacen uso del láser y que tienen como principio de funcionamiento el utilizado en las máquinas fotocopiadoras normales. En este caso, la impresión es muy rápida. Su mayor aplicación se encuentra en la reproducción rápida de páginas con gráficos de muy alta resolución, como las obtenidas en sistemas para diseño asistido por ordenador (CAD). En modelos más evolucionados, desde luego, la impresión es a colores. Estas últimas son, hasta ahora, objetos de lujo incluso para un ordenador personal profesional.

## Impresoras de impacto

Llegamos ahora a la tecnología más tradicional y difundida.

Una impresora de impacto proporciona documentos en papel (con posibilidad también de copia automática, para tener un original y una o más copias inmediatas), utilizando el probado principio de la máquina de escribir, que marca el carácter sobre el papel golpeando el «martillo» con el relie-

ve del carácter sobre la cinta entintada. Los primeros modelos de estas impresoras no eran otra cosa que máquinas de escribir controladas por un interface electromecánico especial. Es evidente que, habida cuenta de las masas de los martillos y del recorrido que deben realizar cada vez (desplazamiento hacia la cinta y posterior retroceso), la velocidad máxima admisible no puede ser elevada y un valor de 15 caracteres por segundo es bastante normal.

Si en lugar de los martillos se utiliza la bien conocida bola, como en las máquinas Olivetti o IBM, por ejemplo, la velocidad puede aumentar hasta más del doble, con la ventaja suplementaria de poder sustituir los tipos de caracteres cuando se quiera y poder así conseguir documentos con una apariencia verdaderamente profesional.

Por último, si se emplea la llamada «margarita», tendremos una nueva posibilidad de aumentar la velocidad, ya que la masa de la cabeza de margarita es todavía menor; en cualquier caso, no se puede superar, el límite de 50 cps (caracteres por segundo). En la Figura 3 se muestra el sistema de impresión de margarita. Esta «flor» gira para llevar el carácter a imprimir frente al vástago de percusión. Con el fin de ahorrar tiempo, los caracteres en los «pétalos» de la margarita están agrupados de modo que los más utilizados están próximos entre sí. Además, la margarita gira durante el desplazamiento horizontal del soporte que, a su vez, se mueve con una velocidad bastante más elevada cuando hay espacios en el texto, es decir, cuando no se debe imprimir nada. Finalmente, la técnica de impresión por impacto incorpora una optimización adicional: en las impresoras de margarita o de bola de alta calidad no se llega al final de la línea si la última parte sólo está constituida por espacios (espacio = ningún carácter). Todo ello hace ahorrar mucho tiempo y se pueden imprimir hasta 70-90 cps (caracteres por segundo).

Cambiando la margarita (o la bola) se obtienen diferentes conjuntos y estilos de caracteres.

Los tipos de impresora que acabamos de describir se utilizan principalmente para trabajos de oficina y, en general, usando se desea tener un documento de calidad. Como se dice en la jerga se consideran impresoras de «letter-quality» (calidad de escritura). Esta denominación se deriva de que los caracteres impresos son bastante nítidos, puesto que



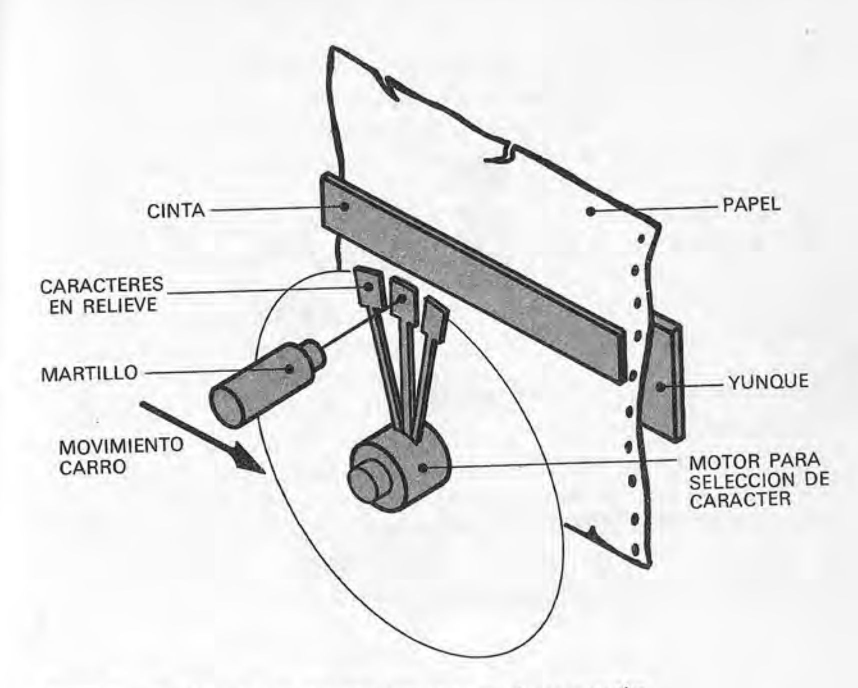



*Fig. 3 - Mecánica de una impresora de margarita.*

cada uno de ellos se consigue gracias a la acción de un relieve (martillo o bola) que incide sobre la cinta entintada. Si, además, consideramos el empleo, muy difundido ya, de cintas en mylar de tipo «usar y tirar», se comprenderá que los caracteres llevados sobre el papel son tan nítidos que parecen impresos en tipografía y, por consiguiente, el resultado es altamente profesional. Como es lógico suponer, también los costes son bastante elevados, por lo tanto, estas máquinas están destinadas sobre todo a profesionales, que las utilizarán en conjunción con sofisticados sistemas de tratamiento de textos.

Cuando, no obstante, el factor «velocidad de impresión» prevalece sobre la calidad de escritura, es preciso pasar a otras tecnologías de impresión, tal como la impresión de impacto con cabeza de agujas que, como veremos más adelante, permite nuevas prestaciones: semigráficas o gráficas.

En la Figura 4 se ilustra la disposición de una típica cabeza de agujas. En la práctica, la cabeza es el soporte don-

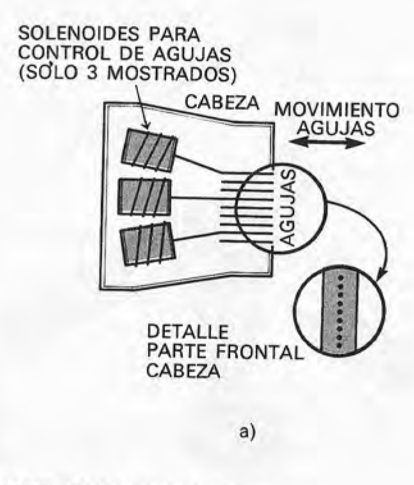






*Fig. 4 - Cabeza de agujas (a) y matriz de puntos (b) que se puede emplear al desplazar la cabeza sobre su guía y accionar las agujas.*

de suelen confluir 7 ó 9 micromartillos, cada uno de los cuales tiene la forma de un pequeño cilindro, tan fino que se le suele denominar aguja. Su diámetro es de unas décimas de milímetro. Cada aguja termina en un cuerpo más consistente que es, en la práctica, el núcleo móvil de un electroimán. Si se excita el electroimán, la aguja será empujada fuera del arrollamiento y sobresaldrá algunos milímetros de la cabeza. Cuando se desexcita el electroimán, un muelle hace retroceder inmediatamente la aguja hacia el interior de su alojamiento en la cabeza.

Nueve agujas están dispuestas en fila vertical, una encima de la otra, preparadas para golpear la cinta entintada



por delante de la cual se desplaza la cabeza. Naturalmente, la percusión se produce solamente cuando las agujas se impulsan hacia afuera. En el papel, arrastrado por un rodillo de goma dura (dentado en los extremos, si se utiliza papel continuo), se imprimen así tantos puntos como agujas activadas existan entre las 9 disponibles. La cinta suele ser de nylon con una tinta poco grasa y la malla es tan fina que cada punto resulta nítido y bien definido.

Ahora estamos en condiciones de estudiar cómo se realiza la impresión de un carácter completo y de las diversas líneas.

La mecánica del arrastre es similar a la de las impresoras térmicas: la cabeza se desplaza en sentido perpendicular al deslizamiento del papel y, durante su movimiento, las agujas se lanzan y recogen, repetidamente. En la Figura 5 se ilustra todo lo anteriormente expuesto. Los caracteres se forman fila tras fila, según un esquema bien preciso que, a su vez, emplea una «matriz» de puntos almacenada en el mi-

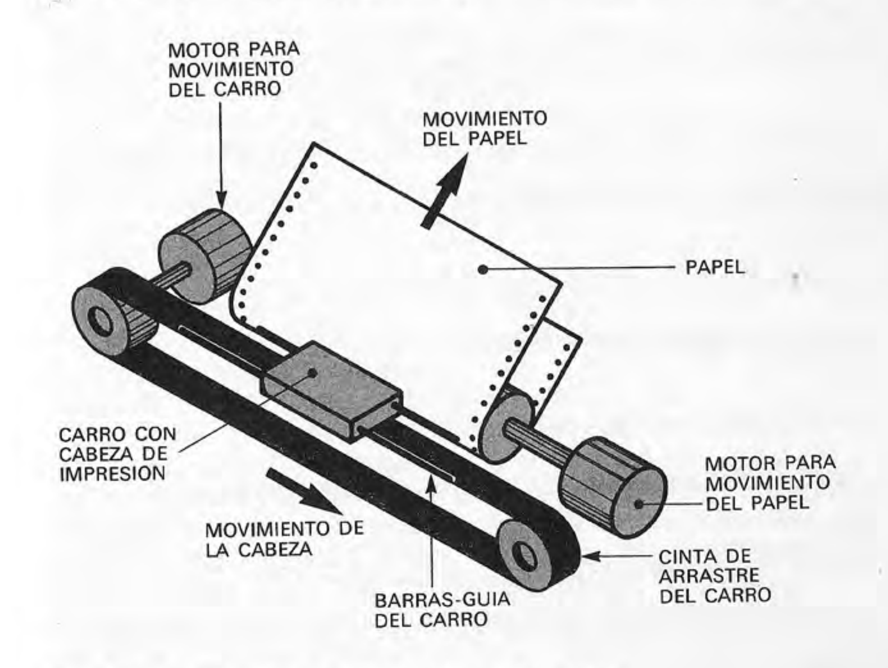



*Fig. 5 - Estructura de una impresora de agujas.*

croordenador que controla la impresora. En cada pasada de la cabeza, de izquierda a derecha (y también de derecha a izquierda, si la impresión es bidireccional), se imprime una línea completa. La longitud de la línea está, pues, limitada solamente por la longitud del soporte de la cabeza.

En los modelos más sofisticados, las líneas de 132 caracteres son lo más habitual, pero es evidente que, aumentando o disminuyendo la velocidad de arrastre de la cabeza, se pueden escribir más o menos caracteres haciéndolos más anchos o estrechos.

Por lo general, la impresión normal es de 10 caracteres por pulgada, la «prolongada» (elongada o evidenciada) de 7 ó 5 caracteres por pulgada, y la «comprimida» de 16 caracteres y medio por pulgada. La Figura 6 muestra algunos ejemplos de impresión, obtenidos con una impresora Epson FIX-80.

¿Qué velocidades se pueden obtener con una impresora de agujas? También muy elevadas: de 300 caracteres por segundo, e incluso mayores, si la impresión está optimizada, es bidireccional y el arrastre de la cabeza se acelera en correspondencia con los espacios. No obstante, en estas im-

IMPRESION NORMAL, 10 C.P.I. (CAR. POR INCH):

ABCDEFGHIJKLMNOPQRSTUVWXYZ - 1234567890

IMPRESION COMPRIMIDA, 16,5 C.P.I.:

ABCDEFGHIJKLMNOPQRSTUVWXYZ - 1234567890

IMPRESION ENFATIZADA, 8,3 C.P.I.:

**ABCDEFGHIJKLMNOPQRSTUVWXYZ - 1234567890**

IMPRESION ENFATIZADA, 5 C.P.I. (CAR. POR INCH):

**ABCDEFGHIJKLMN - 12345**

*Fig. 6 - Ejemplos de impresión con cabeza de agujas.*



presoras nos encontramos con caracteres «punteados», por lo que la nitidez se reducirá. La calidad de escritura proporcionada por una impresora de agujas no podrá ser nunca comparable con la lograda con una máquina de bola o margarita, pero en aplicaciones tales como recibos, facturas, listas, etc., el usuario no busca ciertamente la belleza de un carácter en negrita o en cursiva, sino conseguir que la escritura sea legible y que los impresos o formularios se obtengan con rapidez. Además, hay una ventaja que las de margarita o bola no tienen: las impresoras de agujas pueden utilizarse como impresoras gráficas de puntos con una muy alta resolución en ocasiones. Con programas especiales es posible imprimir los más diversos diseños y figuras y no solamente visualizarlos en pantalla.

Para concluir esta exposición, y antes de examinar la parte electrónica, es obligado citar las impresoras que podemos ver en los «santuarios» de los grandes ordenadores. Por ejemplo, habrá observado que muchos recibos de la luz o del gas tienen caracteres impresos por martillo y no por agujas. Esto significa que se ha empleado una impresora de banda o cadena, capaz de proporcionar hasta 1.500-2.000 líneas de 132 caracteres por minuto.

El procedimiento de impresión, según se ilustra en la Figura 7, es bastante sencillo. Una cadena, que gira con gran rapidez y sin interrupción, lleva en relieve numerosas series completas de caracteres alfanuméricos. El papel se desplaza en sentido perpendicular entre la cadena, la cinta entintada y una serie de 132 vástagos percutores contiguos, con una separación de 1/10 de pulgada. Cuando un carácter que debe estar en una determinada posición pasa por delante del percutor correspondiente, este último se dispara hacia el papel y el carácter queda impreso en el mismo. El procedimiento se repite con rapidez a lo largo de la misma línea, ya que la cadena gira de forma continua y, tal como se ha indicado, lleva varias series de caracteres alfabéticos en relieve, una después de otra. Naturalmente, durante la impresión de cada línea estará bloqueado el arrastre del papel, pero los saltos son tan rápidos que el movimiento parece continuo. Un gran inconveniente de las impresoras de cadena es su alto nivel de ruido: es tan elevado que en los centros de proceso de datos se suelen instalar en recintos aislados acústicamente. No obstante, son tan caras que no

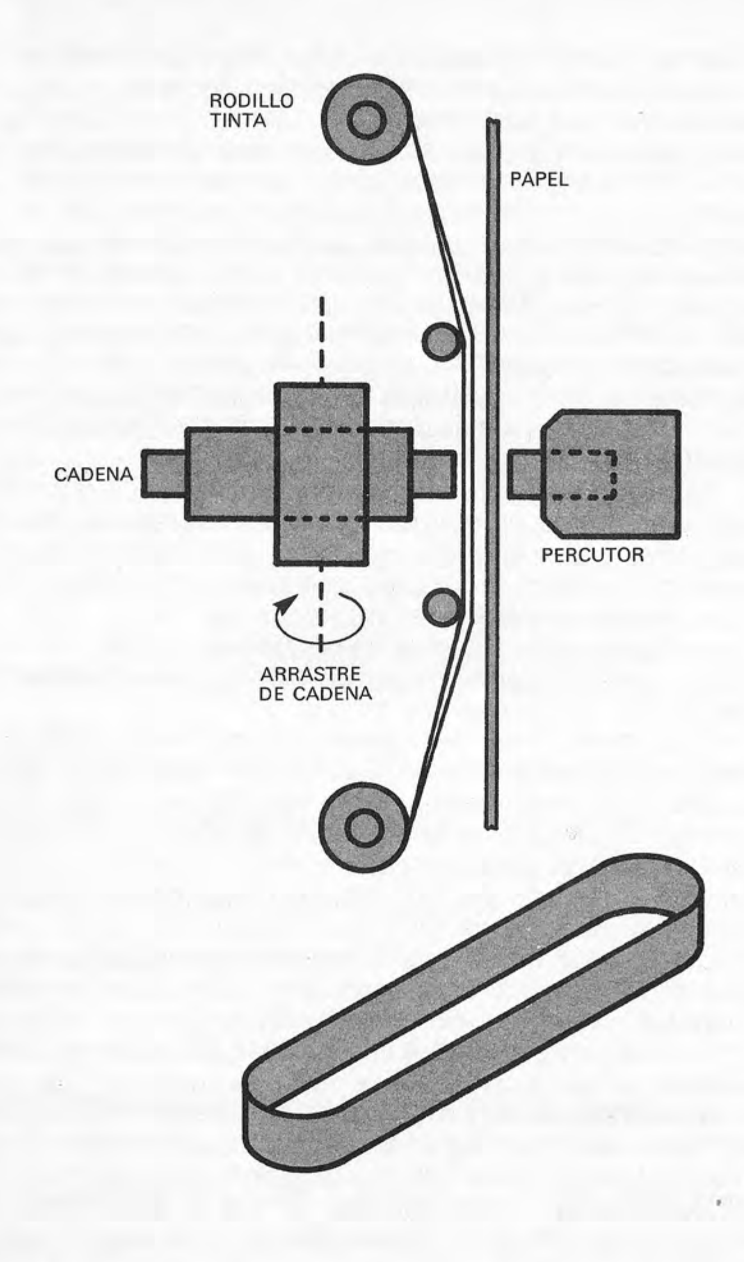



*Fig. 7 - Mecanismo de una impresora de cadena.*



es probable que usted quiera asumir el riesgo de despertar a los vecinos adquiriendo una para su ordenador personal.

## *Conexión al ordenador*

La impresora es uno de los periféricos denominados «inteligentes» porque su complicada parte mecánica está controlada por un pequeño, pero potente, microordenador. Este trabaja de modo que el programa que se ejecuta después de la conexión es único y no modificable; su único cometido es controlar todas las funciones de impresión y, naturalmente, aceptar las órdenes y datos del ordenador, al que está conectada la impresora.

En definitiva, nuestro ordenador personal, en la fase de impresión, no hace otra cosa que transmitir órdenes y datos, en una sucesión muy rápida, al microordenador que se encuentra en el periférico.

Esta comunicación se produce fundamentalmente a través de dos «ports»: uno de salida (en nuestro ordenador personal) y el otro de entrada (en el periférico). En la Figura 8 vemos con detalle el diagrama de bloques de una impresora de agujas típica. El recuadro central encierra prácticamente toda la parte electrónica, constituida por el microor-

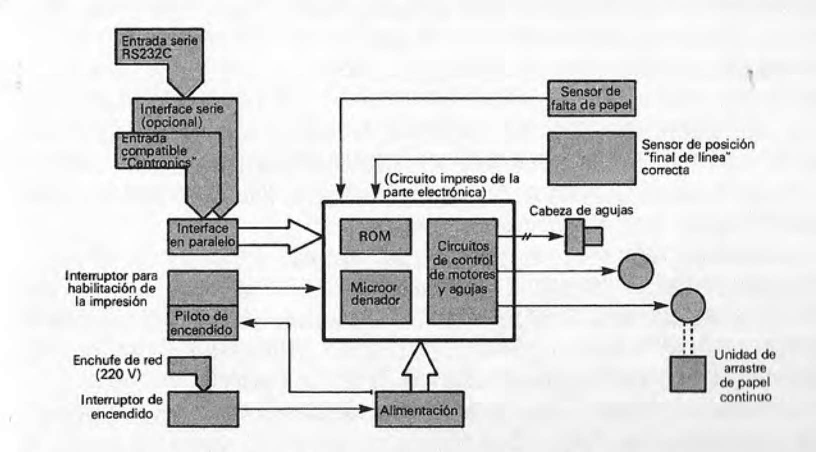



*Fig. 8 - Diagrama de bloques de los elementos que constituyen una impresora de agujas.*

denador y la parte de control de los elementos de potencia: motores y bobinas de las agujas. Los motores son dos, frecuentemente del tipo «paso a paso».

Uno de ellos controla el avance del papel, haciendo girar el rodillo tractor, y el otro actúa sobre una correa que arrasta la cabeza de impresión a lo largo de un soporte horizontal. Una vez que se enciende la impresora, el programa almacenado en la memoria ROM de su microordenador produce una inicialización general: agujas retraídas, cabeza retenida y al final o principio de su carrera, rodillo parado. El µP de la impresora queda entonces atento al «port» de entrada esperando «novedades». Cuando el ordenador personal inicia una impresión comienza a transmitir una secuencia de códigos, cada uno de los cuales tiene un significado bien preciso en la codificación ASCII (vea el capítulo anterior). El µP existente en la impresora está programado parareconocer dicho códigos ASCII, de forma que puede tener lugar la transferencia de los datos hacia el periférico y su interpretación. Carácter tras carácter, un programa que se está ejecutando dentro de nuestro ordenador personal lee el contenido de la memoria que queremos imprimir y transmite todo al exterior através de su «port» de salida, al cual está conectado el cable de la impresora. Los datos, en paralelo o en serie (dependiendo del tipo de conexión elegida entre el ordenador personal y el periférico), llegan uno tras otro al port de entrada del µP que controla la impresora, el cual los lee e introduce en memoria. No obstante, la comunicación no es continua, ya que la transmisión de un carácter entre el ordenador personal y el periférico se realiza de forma mucho más rápida que la posterior impresión; la dificultad se resuelve programando la impresora para que admita cada vez tantos caracteres como pueden estar,como máximo, en una línea de impresión.

En definitiva, la secuencia es la siguiente: 1) el ordenador personal toma un carácter de su memoria y lo escribe en el port de salida 2) espera a que la impresora le de la respuesta «OK, he tomado el carácter; ¡mándame otro!» mediante el envío de una señal ACK y 3) vuelve al punto 1.

Para la impresora, la secuencia es: 1) espera un carácter procedente del ordenador personal 2) cuando llega el carácter (en ASCII), lo lee en el port de entrada y lo introduce en memoria 3) avisa al ordenador personal: «he tomado el dato, ¡mándame otro!» 4) la secuencia se repite hasta completar el número de caracteres que haya en cada línea 40, 80, 132 enviando entonces al ordenador una señal de «ocupado» -5) se imprime la línea completa (en tanto que el ordenador personal sigue esperando que la impresora quede libre y, por tanto, no transmite) y 6) terminada la impresión de la línea, la impresora avisa al ordenador personal de que está de nuevo preparada y la secuencia vuelve al punto 1.

Cuando la impresora tiene en memoria, bajo la forma de un bloque de códigos ASCII, toda la línea a imprimir, el microordenador que la controla se dedica completamente al control de los motores y de las agujas de la cabeza. Esta última comienza a ser arrastrada de izquierda a derecha y, durante su recorrido, las agujas se lanzan de modo que formen en el papel, mediante puntos, los caracteres correspondientes a los códigos anteriormente almacenados en memoria.

Si se fija en la Figura 4, podrá ver cómo se forma un carácter: basta golpear durante el movimiento todos los puntos que componen el contorno del carácter, seleccionándolos en el momento oportuno entre los 6 × 9 disponibles. La línea se imprime en una sola pasada, ya que tenemos 9 agujas en columa, una encima de la otra. Al finalizar la impresión de la línea, el microordenador detiene el desplazamiento de la cabeza y hace avanzar el motor de arrastre del papel una línea hacia arriba. Así es posible iniciar la impresión de la línea siguiente, una vez que esté cargada en memoria.

La impresión puede producirse al ir de izquierda a derecha sólo (en cuyo caso, ha de llevarse la cabeza al final de su recorrido) o bien se imprime también al ir de derecha a izquierda (procedimiento típico de los modelos más perfeccionados). En tal caso, la impresión se denomina bidireccional y, al eliminar el retorno al principio de nueva línea, lavelocidad aumenta de forma considerable.

También es fácil comprender que si en vez de controlar las agujas con el único objeto de imprimir caracteres con puntos previamente determinados, se controlan de forma individual, se podrían generar todas las figuras y diseños quedeseemos, incluso los constituidos por secuencias de puntos. Naturalmente, se precisarán más pasadas y la impresión

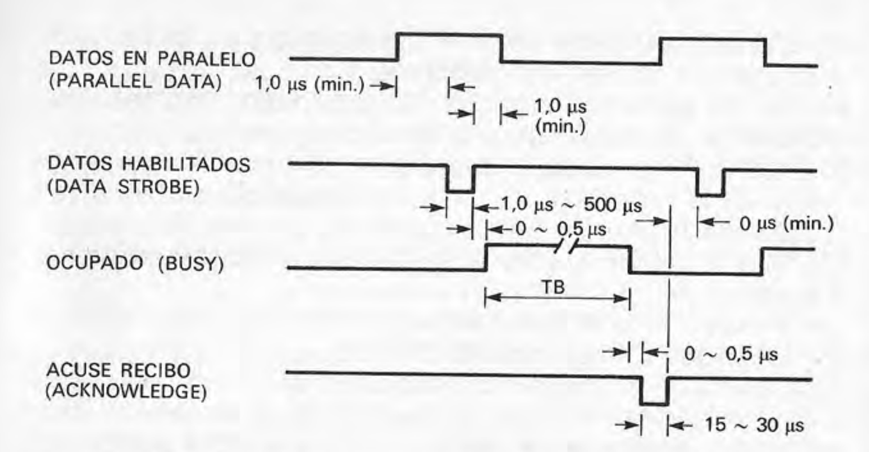



*Fig. 9 - Señales de comunicación entre el ordenador y la impresora (interface paralelo).*

será más lenta, pero podremos conseguir óptimos resultado gráficos, representando en el papel diagramas de barras, proyecciones ortogonales, dibujos en perspectiva, imágenes, etc. En tal caso, se dice que la impresora tiene incorporada la opción gráfica.

En la Figura 9 se muestra un ejemplo de las señales de comunicación adoptadas entre el ordenador personal y una impresora, si la conexión se realiza através de un interface paralelo, mientras que para el caso de que este interface sea compatible con el Centronics (normalizado) la figura 10 contiene la descripción de todas las líneas típicas de control.

## Cómo seleccionar una impresora

Para concluir esta visión panorámica de las impresoras, sería conveniente ver algunos criterios de selección. Ante todo, hay que basarse en el tipo de actividad a que se aplicará: si el procesador está destinado a utilizarse sobre todo para desarrollar software de aplicación (programas concebidos para otros ordenadores o para la automatización industrial, por ejemplo), será útil una impresora muy rápida, con posibilidades gráficas, siendo la más indicado una de impacto con cabeza de agujas. Si la impresora se emplea



| Señal | N.º* patilla | Tipo para la impresora | Descripción |
|---|---|---|---|
| Data Strobe (Datos habilitados) | 1 | Entrada | Llega desde ordenador para indicar a la impresora que ha entrado un dato. |
| Data Bit 1<br>Data Bit 2<br>Data Bit 3<br>Data Bit 4<br>Data Bit 5<br>Data Bit 6<br>Data Bit 7<br>Data Bit 8 | 2<br>3<br>4<br>5<br>6<br>7<br>8<br>9 | Entrada<br>Entrada<br>Entrada<br>Entrada<br>Entrada<br>Entrada<br>Entrada<br>Entrada | 8 bits del dato enviado por el ordenador |
| Acknowledge (Acuse recibo) | 10 | Salida | Indica al ordenador que el carácter se ha recibido de forma normal |
| Busy (Ocupado) | 11 | Salida | Indica al ordenador que la impresora está ocupada |
| Paper out (agotam. papel) | 12 | Salida | Indica falta de papel |
| Select | 13 | Salida | Indica si la impresora está, o no, activa |
| O V | 14<br>16<br>33 | E/S | Masa de señal |
| CHASSIS GROUND | 17 | E/S | Masa chasis |
| + 5 V | 18 | Salida | Alimentación suplementaria 50 mA |
| O V | 19 a 30 | E/S | Retorno masa señales |
| * Del conector | | | |


*Fig. 10 - Líneas del interface paralelo compatible con Centronics.*

en trabajos de oficina, o donde el ordenador personal tenga funciones típicamente de gestión para la obtención de documentos, puede ser conveniente una impresora de buena calidad de escritura: de bola o de margarita; lo mismo puede decirse con respecto a aquellas aplicaciones en las cua-




*Foto 4 - La conocida impresora EPSON FX-80.*

les se imprimen los resultados de procesos de textos, correspondencia automatizada, etc. Si se comienza a trabajar con un ordenador personal, en una fase en la que las inversiones estén limitadas, puede ser conveniente la adquisición de una impresora térmica o una electrostática. Para aplicaciones en el campo industrial son recomendables las impresoras térmicas o modelos con cabeza de agujas y un número limitado de columnas (un máximo de 20 a 25), porque el espacio ocupado debe ser mínimo.

Los precios varían no siempre de modo proporcional a las prestaciones de la máquina, aunque puede decirse que los costes más moderados son «patrimonio» de los modelos de agujas; ésta es la razón de que sea el tipo de impresora más utilizado como periférico para los ordenadores personales actuales.

## GLOSARIO

PAPEL CONTINUO: Banda de papel cuyos bordes están perforados a intervalos determinados y en correspondencia con los dientes existentes en los extremos del rodillo portapapel de la impresora, donde pueden introducirse y «hacer presa» para asegurar un arrastre seguro del papel. Cada 28 cm aproximadamente, el papel tiene un pliegue, que facilita también la separación de hojas individuales. Así plegado y empaquetado «en forma de acordeón» el papel continuo se suministra en paquetes de 2.000 hojas o más.



BASE DE DATOS («Data Base»): Sistema de datos, ordenados según secuencias bien determinadas definidas a priori por el usuario, que reside en la memoria del ordenador (en disco o en RAM). Con las oportunas órdenes del sistema operativo, el usuario puede seleccionar y extraer los datos que poseen características similares, tales como: «direcciones de todos los clientes cuyo apellido comience por <BRA>», «números de todos los recibos emitidos el 15/06/1984», etc. Quienes trabajan con bases de datos y con los programas que las gestionan (DBMS, Data Base Managment System-Sistema de Gestión de Base de Datos) siempre tendrán necesidad de una impresora para el volcado («dump») de los datos buscados.

DUMP: Volcado del contenido de una zona de memoria en otro soporte físico, tal como un disco (volcado en disco) o una hoja (volcado en papel).

TERMINAL: Periférico de un ordenador generalmente constituido por un teclado alfanumérico, un monitor de video (que suele ser también gráfico) y un hardware de control, naturalmente basado en un microprocesador. Este último permite la conexión (frecuentemente en modo serie) al ordenador con el cual el usuario desea «dialogar». En la mayor parte de los ordenadores personales existentes en el mercado, el terminal está integrado en la estructura de base del sistema y, por consiguiente, no se le considera un periférico externo propiamente dicho.

CPS: Caracteres (impresos) por segundo. Valores característicos para las impresoras de agujas son: 80, 100, 120, 160 y hasta 3.000 cps.

CPI: Caracteres (impresos) por pulgada (inch). Medida de lo compacta que es la impresión. Cuantos más caracteres hay en una pulgada tanto más comprimida será la impresión y viceversa. Los distintos valores de com-

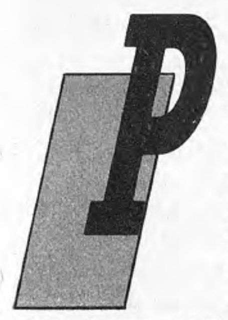
prensión se obtienen desplazando la cabeza a diversas velocidades mientras que las agujas son lanzadas a la misma cadencia de una impresión normal; cuanto más rápidamente se mueva la cabeza, tanto menos comprimida será la impresión y tanto más bajo será el valor de CPI.

MOTOR PASO A PASO: Se trata de un motor especial de corriente continua que no tiene un movimiento giratorio uniforme, sino que avanza a saltos, aunque pequeñísimos. Puesto que después de cada salto el motor puede bloquearse en una posición estable, su empleo está especialmente recomendado en todas las aplicaciones para máquinas en las cuales los avances y los desplazamientos deben ser discretos es decir, a saltos, y precisos.



# CAPITULO VII

## *MEMORIAS DE MASA Y OTROS PERIFERICOS DE ENTRADA/SALIDA*



PARA su correcto funcionamiento, un ordenador debe tener necesariamente, además de la CPU, una cantidad de memoria adecuada. Hemos visto en los capítulos anteriores que parte de esta memoria debe ser ROM (permanente) y otra parte RAM (alterable por la CPU). Cuando deja de aplicarse la alimentación, la memoria RAM perderá todo su contenido, por ello es preciso dotar siempre al sistema de un medio para "salvar" los datos de la memoria de modo que se puedan recuperar luego cuando se vuelva a encender la máquina. Se utilizan con este fin periféricos denominados "memorias de masa".

Existen muchas técnicas para copiar el contenido de la RAM en el exterior del ordenador, pero las más empleadas utilizan soportes magnéticos tales como cintas de casete y discos. Estos últimos pueden ser flexibles o rígidos (también llamados duros). El volcado de los datos de la memoria en el soporte magnético se efectúa a través de un interface especial que suele estar constituido por chips del tipo de integración a muy alta escala (VLSI). Un programa, residente en la memoria ROM del ordenador, extrae de la memoria un dato cada vez y lo transmite al chip de interface, que "se ve" desde la CPU ni más ni menos que como un "port" normal de E/S. Este chip, que suele denominarse controlador (de discos o de cinta) genera señales eléctricas mo-

duladas que dependen exactamente de los "unos" y de los "ceros" del dato binario recibido y que llegan, a través de un cable blindado especial, a una cabeza de grabación bajo la cual se desliza el soporte magnético (cinta o disco). Sea cual sea, el soporte se magnetiza y, al final de la grabación, conserva una copia de los datos que están en memoria. Cuando queramos devolver los datos a la memoria central, bastará que el chip controlador lea el soporte magnético con la cabeza de lectura (que es la misma utilizada en la fase de grabación), descifre las señales eléctricas extraidas y coloque el dato resultante en el port para que lo lea el ordenador.

Hay tres tipos de soportes magnéticos muy difundidos. El primero es la cinta de casete, para la cual empleamos una grabadora normal; el proceso es lento pero resulta muy económico.

El segundo método emplea como soportes discos magnéticos hechos del mismo material que la cinta. Se denominan discos flexibles o disquetes ("floppy-disk" en inglés). Para su escritura y lectura se utilizan unidades especiales, denominadas unidades de disco. Son algo más complejas que una grabadora de casete, pero mucho más rápidas (hasta 200 veces más). Ofrecen, además, la gran ventaja del denominado acceso directo, que describiremos más adelante. Naturalmente, el coste es bastante más elevado, aunque en conjunto se compensa si se tiene en cuenta la cantidad de información que puede grabarse en un disco flexible en comparación con una cinta de casete normal.

El tercer sistema de almacenamiento masivo utiliza los llamados discos duros o rígidos ("hard-disk", en inglés). Se parecen en su aspecto a los discos flexibles, pero son de material metálico y están envueltos enn fundas especiales herméticas. Un disco rígido permite almacenar hasta 50 veces más datos que un disco flexible de las mismas dimensiones, lo cual es posible por cuanto que la ausencia de polvo y la precisión del soporte metálico permiten grabar la información de un modo mucho más compacto.

## Accesos secuencial y aleatorio

Una distinción importante en el campo de las memorias se basa en el método de acceso, es decir, en la forma mediante la cual el ordenador puede extraer de ellas la información. Se habla de acceso "secuencial" y de acceso "directo" o "aleatorio" (en inglés "random"), métodos característicos de las cintas y de los discos, respectivamente.



En el caso del acceso secuencial si queremos leer (o escribir) los datos situados en una posición, y nos encontramos en otra, tenemos forzosamente que recorrer y leer todos los datos intermedios, aún cuando no nos interesen en absoluto. Esto se ve muy claro en el medio típico de acceso secuencial: la cinta.

El caso contrario es, por ejemplo, el de los discos flexibles: su superficie está subdividida en pistas circulares concéntricas (esto es, no se trata de una sola pista contínua en espiral, como en los discos musicales) y cada una de estas pistas está, a su vez, dividida en sectores (normalmente, en un disco flexible de 8 pulgadas —20,32 cm— se tienen 77 pistas por cara, cada una con 8 sectores capaces de almacenar cada uno unos 250 bytes). Cuando queremos acceder a un sector particular, el brazo portacabezas se desplaza hasta situarse sobre la pista correspondiente y espera la llegada del sector (olvidábamos decir que, evidentemente, el disco se pone a girar cada vez que se realiza una operación de lectura/escritura) pasando así sobre el mínimo e indispensable número de datos que no interesan. Con ello se aumenta enormemente la velocidad y, sobre todo, la flexibilidad con que se realizan las operaciones de E/S, lo que resulta fundamental en la gestión de los ficheros (archivos de datos) y en el funcionamiento de los sistemas operativos. Gran parte de estos (los "serios") se conocen con la abreviatura DOS (Disk Operatig System-Sistema Operativo de Disco) porque controlan la unidad de disco y se guardan en parte en éste, hasta que se necesitan (momento en el cual se pasan a la memoria central).

A propósito de los discos, otro punto importante es el denominado "formateo". Se trata de la escritura preliminar de señales adecuadas en el disco (antes de que lo utilicemos). Sirven fundamentalmente para dos cosas: 1) crear puntos de referencia (como surcos magnetizados) y de sincronismo en la lectura/escritura, y 2) definir la posición de comienzo de los datos propiamente dichos. Esta operación, lamentablemente diferente de un ordenador a otro, es una de las primeras cosas que se aprende a hacer con un ordena-

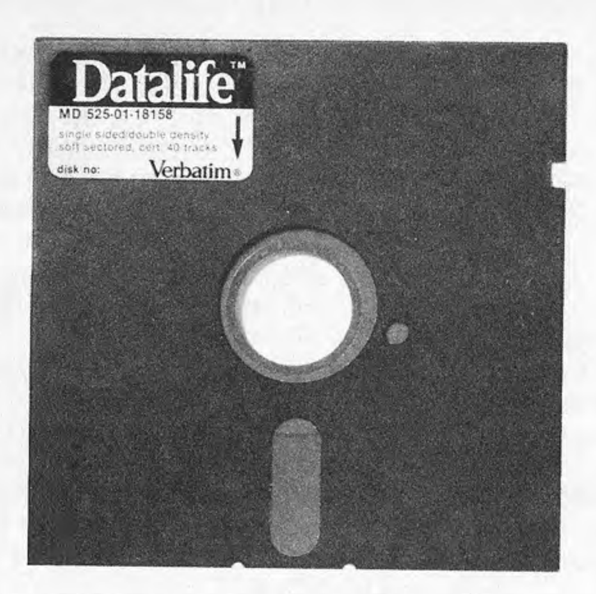



*Foto 1 - Disco flexible de 5 1/4 pulgadas.*

dor personal. Roba un poco de espacio a los datos y por ello será preciso distinguir la capacidad de un disco flexible "formateado" o "no formateado" (contando únicamente la primera para los fines prácticos).

Cada sistema debe tener el periférico de memoria de masa que mejor se adapte a sus dimensiones y a su grado de sofisticación: un ordenador personal para actividades caseras, por ejemplo, está provisto de interface para grabadora de casete en la versión básica, y sólo con carácter opcional se le podrá dotar de una unidad de disco flexible. Por el contrario, un ordenador personal evolucionado, utilizado en aplicaciones de gestión o científicas, se suministrará con una estructura de base que comprenderá al menos una unidad de disco, puesto que, de no ser así, sería imposible trabajar de un modo serio y, sobre todo, productivo. Asimismo, los modelos más sofisticados podrán tener como dotación estándar dos unidades de disco o bien, una unidad de disco rígido, con gran capacidad y velocidad. La ventaja de una memoria de masa rápida no es solamente la de poder "salvar" todo lo que haya en el ordenador antes de su desconexión, sino también la de permitir una gestión ágil de






*Foto 2 - Microdisco flexible de 3 pulgadas y media en cartucho de plástico.*

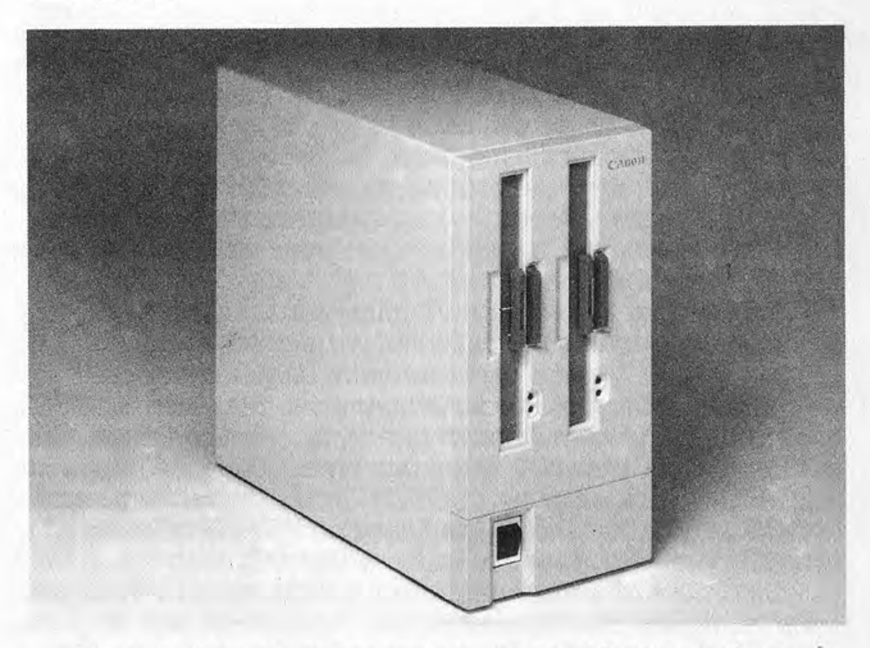


*Foto 3 - Conjunto de dos unidades de disco flexible de Canon.*



*Foto 4 - Unidad de disco rígido en la que puede verse el disco al haber quitado la tapa.*

los recursos de software del ordenador, utilizando los discos mientras se desplazan bloques de datos, programas, procesos parciales de cálculo, etc.

Para aprovechar las unidades de disco es necesario que el sistema operativo del ordenador sea sofisticado, rápido y potente al mismo tiempo, y capaz de controlar cualquier transferencia de datos, desde o hacia los discos, de la manera más sencilla posible.

Un buen sistema operativo para el control de los discos es quizá en ciertas circunstancias lo que más condiciona al usuario en la elección de la máquina. Sistemas operativos de gran difusión y prestaciones comprobadas son: el CP/M (Control Program por Microcomputers) para Unidades Centrales de Proceso (CPU) tales como Z80, 8080, 8085 y 8086/8088 (para estas es el CP/M-86 de Digital Research), el DOS 3.3 y el Pro-DOS para Apple, el PC/MS-DOS (de Microsoft) para el 8088 del ordenador IBM-PC, etc.

Un Sistema Operativo de Disco (DOS) debe integrar una serie completa de órdenes fáciles y potentes para el control de la memoria de masa con otra igualmente potente serie de órdenes para manejar, de la mejor manera posible,



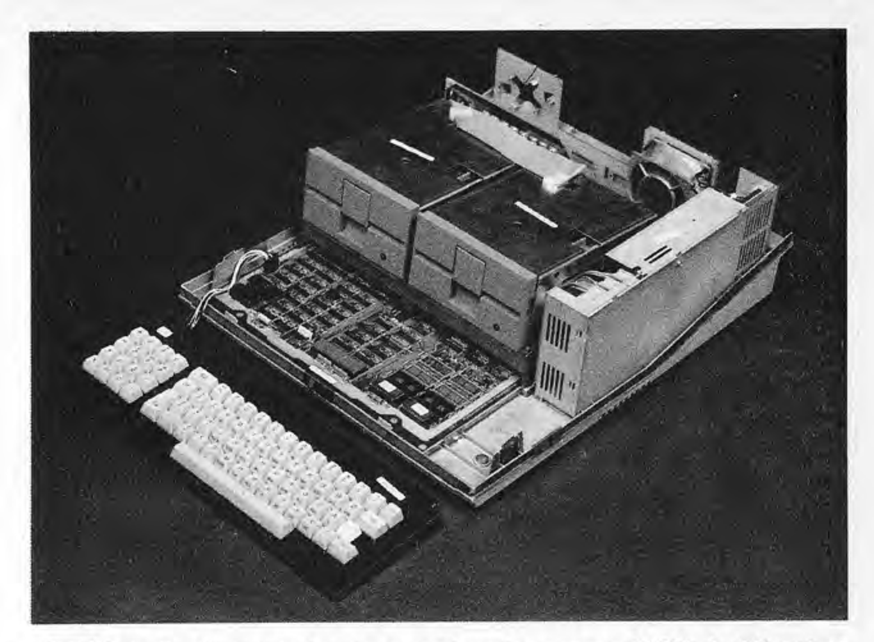


*Foto 5 - Vista interior de un ordenador personal M20 de Olivetti; se observan las dos unidades de disco flexible de 5 1/4 pulgadas (que están integradas en el sistema) en primer plano.*

los recursos internos del ordenador: espacio de memoria, presentación visual de los datos y/o de los programas, ejecuciones simultáneas de varios programas, etc.

Asimismo también es verdad que la fiabilidad de los propios periféricos de memoria de masa está en estrecha relación con las buenas características globales del ordenador personal: cuanto más se desciende hacia las actividades caseras, tanto más se tiene que hacer frente a evidentes compromisos entre prestaciones y coste. Tenga siempre presente que una vez apagada la máquina, debe tener la completa seguridad de poder volver a leer su casete o sus discos sin errores, puesto que, de no ser así, habría perdido tiempo y dinero.

Finalmente, le damos una recomendación que, por propia experiencia, le aconsejamos siga al pie de la letra: si se decide por la compra de una máquina fiable y de buenas prestaciones, no busque ahorrarse algunas pesetas adquiriendo soportes magnéticos de poca calidad, al final le resultarán más caros.

Como ilustración de nuestra exposición sobre las memorias de masa, las fotografías 1, 2, 3, 4 y 5 muestran el aspecto que presentan un disco flexible, tanto de 5 1/4 pulgadas como de 3 pulgadas y media, un periférico típico que contiene dos unidades de disco flexible, una unidad para disco rígido y finalmente, dos unidades de disco integradas en la estructura básica del ordenador personal M20 de Olivetti.

## *Otros periféricos: a) periféricos "de salida"*

Agrupamos en esta categoría a todos los periféricos que permiten al sistema enviar sus datos hacia el mundo exterior, por lo que a dicha categoría pertenecen: las impresoras, las pantallas de texto y/o gráficos, los dispositivos trazadores ("plotters") y los interfaces musicales y vocales. Hagamos una descripción rápida de las unidades más significativas, excluyendo las impresoras, que han merecido un apartado propio.

## *Plotter*

Un plotter o trazador de gráficos es una máquina capaz de dibujar cualquier tipo de figura por medio de una especie de lápiz que se desplaza sobre una hoja de papel, normalmente fija. El lápiz es controlado, en su movimiento, por dos motores: uno que lo puede desplazar a lo largo de la dirección del eje x, haciendo que se mueva el brazo portalápiz; y otro que lo mueve en la dirección "y", perpendicular a la "x", sobre el propio brazo, por lo que el lápiz puede llevarse con facilidad a un punto cualquiera del plano x-y. En la fotografía 6 se muestra el aspecto de un moderno plotter (M84 de Calcomp), cuya parte electronica de control está gobernada por un microprocesador incorporado.

Cuando una persona dibuja, desplaza su mano controlando el movimiento con los ojos, realizando una comparación de cada posición con la imagen, prácticamente acabada, que tiene en su cerebro. Lo mismo sucede en el caso del plotter: el lápiz se lleva, en cada momento, a una posición de la hoja de papel cuyas coordenadas se envían des-





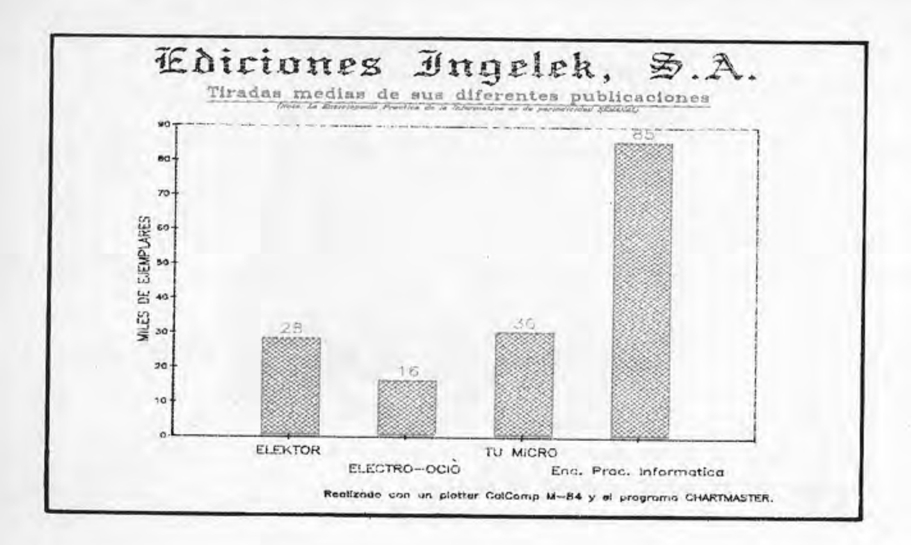

*Foto 6 - Plotter M84 de Calcomp.*

de el ordenador al que está conectado, y que tiene en su memoria el dibujo traducido ya a puntos. El dibujo se consigue, pues, con una serie de trazos infinitesimales, que corresponden cada uno al rastro dejado por el lápiz en su recorrido desde un punto a otro muy próximo. Para una realización "limpia" de los dibujos es necesario que el ordenador que transmite las órdenes al plotter posea un software bastante sofisticado.

El plotter es un periférico con prestaciones muy espectaculares y que pone a disposición del usuario del ordenador personal grandes posibilidades gráficas que no pueden proporcionar la pantalla ni la impresora por sí solas. Lamentablemente, es todavía una unidad de relativamente alto coste, aunque, con la evolución de la tecnología y la consiguiente reducción de los precios, existe la expectativa de que, en un futuro no muy lejano, se produzca la aparición en el mercado de dispositivos con grandes prestaciones y unos precios más moderados. En la fotografía 7 se ilustra algunos ejemplos de dibujos realizados con el plotter M84.

## *Terminales*

La visualización de datos en forma de gráficos o, más simplemente, de caracteres alfanuméricos, imprescindible



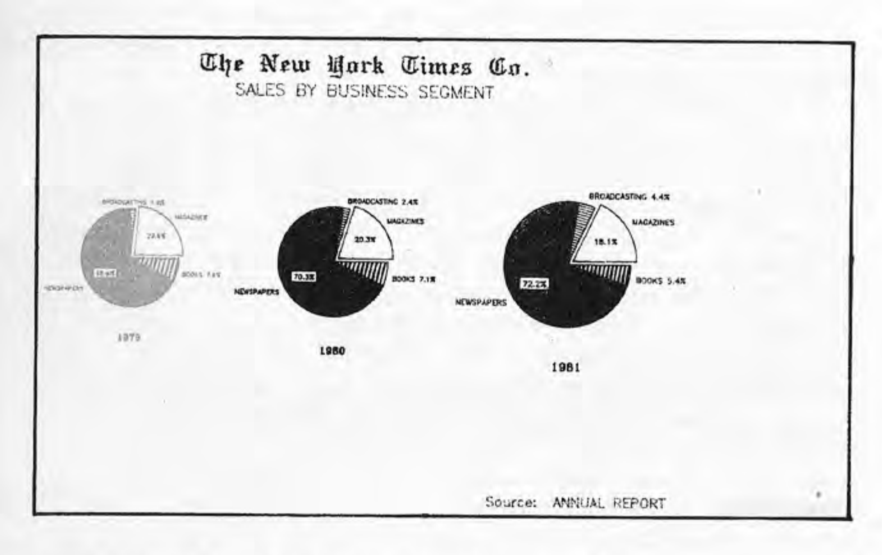




*Foto 7 - Algunos gráficos obtenidos con el M84.*



en las aplicaciones informatizadas se logra mediante periféricos bastante sofisticados que se denominan "terminales". Un terminal es un periférico tanto de entrada como de salida, puesto que admite también datos a través del teclado alfanumérico.

La parte del terminal que se encarga de la entrada desde el teclado es banal y poco sofisticada (como se pone de manifiesto en el sencillo ejemplo de microordenador personal descrito en el capítulo cuarto), mientras que la visualización es una operación delicada y que merece algunas observaciones complementarias.

¿Recuerda cómo funciona un televisor? Pues bien, un terminal es algo muy similar. También en este caso hay un haz de electrones que, al recorrer de una manera preestablecida la pantalla excita el fósforo que la recubre y dibuja las figuras. El haz de electrones, que explora o barre la pantalla en 1/50 de segundo, la recorre en su totalidad mediante unas trazas, que se disponen horizontalmente una debajo de otra (lo que se denominan líneas), que crea en el sentido de izquierda a derecha y de arriba abajo (como leemos la página de un libro). En la figura 1 se muestra una representación esquemática del "pincel" de electrones.

Al final del cuadro, después de haber recorrido 625 líneas, el pincel electrónico vuelve al punto denominado "I", y comienza desde el principio su zigzageante recorrido. Durante los finales de línea y de cuadro, el pincel se apaga, por lo que no aparecen esos trazos "de retorno" en el fósforo que recubre la pantalla; si el pincel electrónico estuviera siempre encendido, veríamos todas estas 625 líneas y los trazos de retorno nítidos, blancos y tan cerca unos de otros que la impresión sería la de un único rectángulo blanco. Pero el pincel puede tomar dos estados distintos: encendido y apagado, además de tener una intensidad regulable. Si un circuito controla el recorrido del pincel electrónico y, mientras tanto, en sincronismo con todas las demás fases de la pesentación visual, lo enciende y lo apaga, se generarán en la pantalla puntos u, ocasionalmente, segmentos. Aproximando de modo controlado estos puntos y/o segmentos, se podrán realizar con facilidad contornos de caracteres alfanuméricos o gráficos. El control del encendido y del apagado del pincel electrónico se realiza por medio de un solo chip integrado VLSI, denominado "controlador de CRT" (por

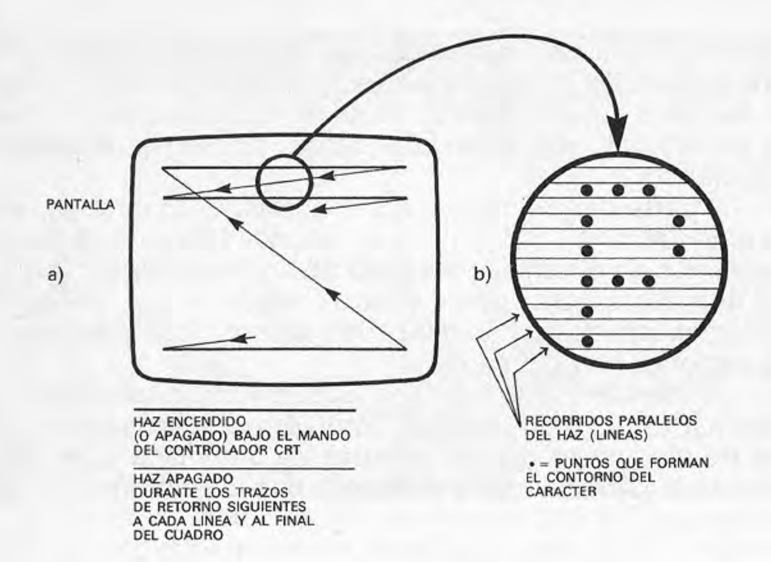
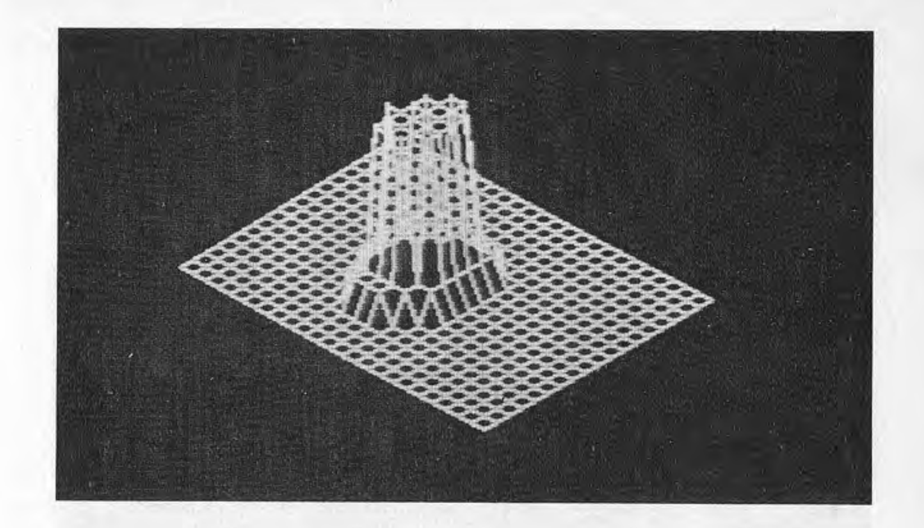



*Fig. 1*
*a) Recorrido del haz de electrones en la pantalla durante un cuadro. Cada segundo se realizan 50 cuadros, constituido cada uno por 625 líneas.*
*b) Detalle de la generación de un carácter en una pantalla de CRT. Los puntos son las posiciones de las diversas líneas en donde el chip controlador de CRT enciende el haz de electrones, por lo que se ven puntos luminosos. El efecto total es el de delimitar el contorno de un carácter alfanumérico o gráfico. En el ejemplo se generó la letra "P".*

Cathode Ray Tube = tubo de rayos catódicos). De este modo la CPU no tiene que preocuparse en absoluto de seguir las fases de la presentación visual y de sincronizarse con las evoluciones del haz electrónico, porque de todo ello se encarga el controlador de CTR, que la CPU "ve" como un port de salida normal. Controladores muy usados son el 6545 de Rockwel (lo usan los equipos con CPU 6502 como el AIM-65 o el Apple) y el 8275 de Intel (lo emplea el IBM-PC).

En la figura 1b se muestra el efecto que se obtiene encendiendo el pincel electrónico solamente en determinadas posiciones de su recorrido. En el ejemplo se genera el carácter "P". Si la CPU sigue transmitiendo nuevos datos al chip controlador, éste generará nuevos caracteres o signos gráficos hasta que se llene toda la pantalla. Entonces, para que los nuevos caracteres no sustituyan, por superposición, a los





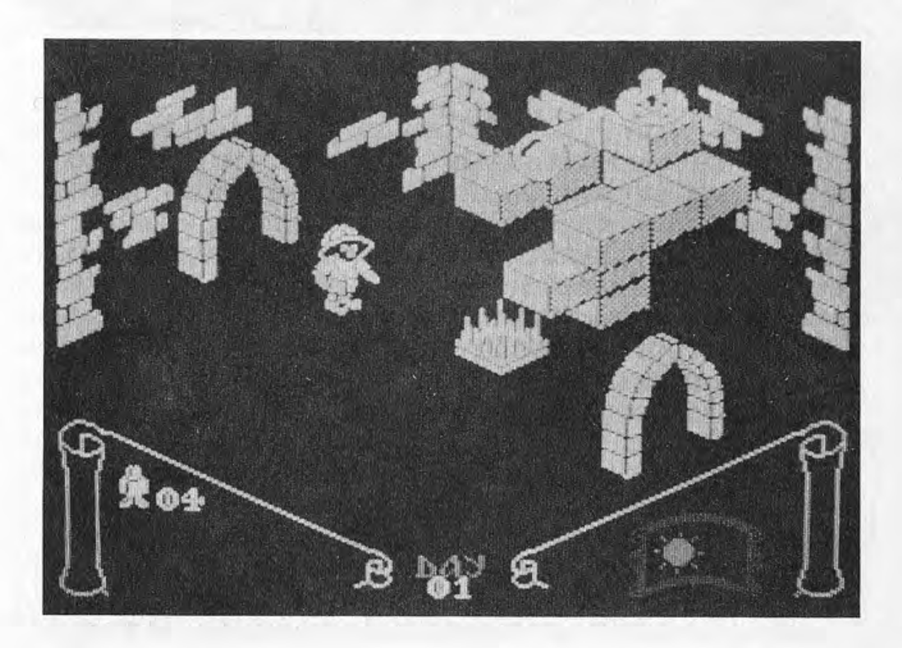


*Foto 8 - Algunos gráficos realizados en pantallas de media resolución.*

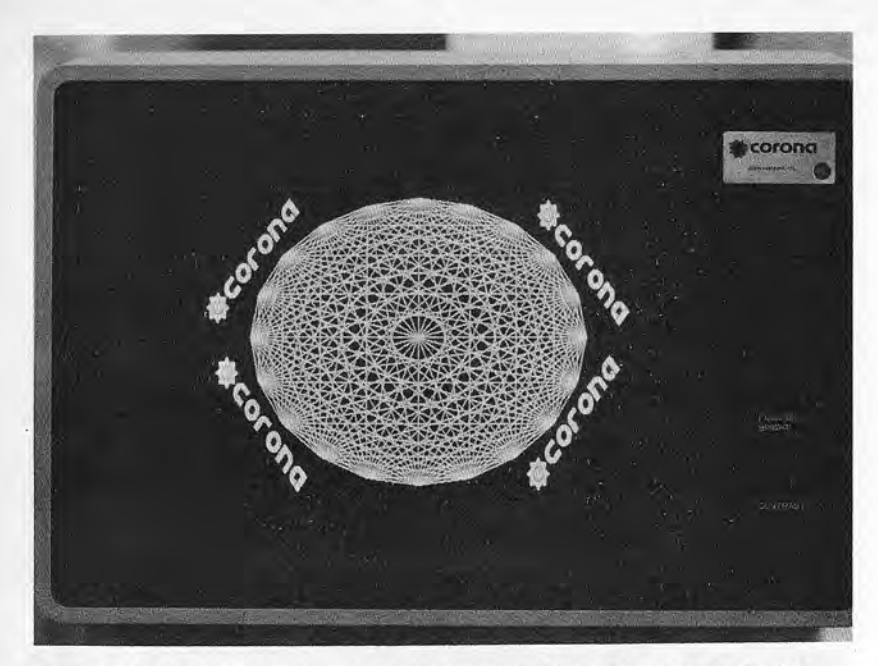



*Foto 9 - Imagen de una pantalla con gráficos de alta resolución.*

antiguos, se puede mandar la orden oportuna al chip controlador de CRT, que irá desplazando hacia arriba las líneas. Algunos terminales gráficos pueden controlar hasta 1024 × 1024 puntos independientes. Puede imaginarse la increible resolución (definición) de las figuras generadas en terminales de esta naturaleza. Por curiosidad, hace solamente tres años se precisaban un mínimo de 50 circuitos integrados diferentes para realizar lo que actualmente se consigue con dos o tres chips... y, naturalmente, las prestaciones son bastante mejores.

En la actualidad hay pocos ordenadores personales provistos de terminal externo; casi todos ellos están provistos de teclado, integrado en la estructura, y de interface interno de vídeo, por lo que suele bastar conectar un monitor (o aparato de TV normal) y todo funcionará de manera inmediata.

En la fotografía 8 se muestran algunos ejemplos de gráficos realizados en pantallas con una resolución media, mientras que en la fotografía 9 vemos la nitidez de las imágenes de alta resolución.





*Foto 10 - Interface de voz para el popular Spectrum.*

## Interfaces de sonido (voz y música)

El ordenador personal ha sido utilizado, desde su introducción en el mercado por los apasionados a la música, debido a sus enormes posibilidades de control, tan rápido que es casi en tiempo real, de los parámetros de los nuevos instrumentos musicales: los sintetizadores. Sin entrar en demasiados detalles, sólo diremos que actualmente muchos instrumentos comerciales están dotados de un interface normalizado para la conexión a cualquier ordenador. Dicho interface no es otra cosa que un conjunto de "ports", tanto de entrada como de salida, que han de conectarse a otros idénticos del propio ordenador.

Olvidábamos decir que los modernos sintetizadores están provistos de un potente sistema electrónico digital con microprocesador, que controla los parámetros de cada sonido, permite la polifonía, arpegios automáticos y una gran multitud de efectos. En el comercio existen accesorios, en forma de tarjetas a insertar en una o varias ranuras del ordenador personal, que lo transforman en un instrumento so-

fisticado: basta unirle un teclado y podremos "tocarlo". Interfaces de esta categoría son las tarjetas Mountain Hardware y el sistema Alpha Synthaur para Apple.

Con el mismo principio de la tarjeta adicional existen interfaces de voz, basados en chips VLSI muy perfeccionados, que traducen un código binario, transmitido desde la CPU, en un fonema: oh, ah, b, ch, por ejemplo. Una serie de órdenes binarias proporciona así una sucesión de fonemas que permiten "oír" una palabra al unirse (la fotografía 10 muestra el modelo de este periférico adecuado para el Spectrum). De esta forma, en plan sibarita, el interface puede leer un texto a la vez que lo imprimimos o visualizamos. ¿Recuerda la película "Juegos de Guerra" y el interface Votrax conectado al IMSAI del protagonista? Los únicos defectos (además del precio, claro) en el estado actual de la tecnología, son la carencia de inflexiones en la expresión y la falta de chips especializados en pronunciaciones diferentes: no hay más remedio que habituarse al fuerte acento inglés.

## *Otros periféricos:* b) *periféricos de entrada*

Un periférico de entrada recoge datos y/u órdenes del mundo exterior y los transmite al ordenador bajo la forma de códigos binarios. Son periféricos de entrada, además de los teclados alfanuméricos, las tabletas gráficas, el joystick, el ratón, los "paddles" y, para aplicaciones sofisticadas, todas las unidades de conversión analógico-digital tales como tarjetas de osciloscopio digital, analizadores de espectro, etc. Estos últimos productos suelen estar disponibles como accesorios insertables en las ranuras de la tarjeta matriz del ordenador personal.

## *GLOSARIO*

FLOPPY DISK: Disco flexible de material plástico, sobre el cual está depositado un estrato de óxido metálico magnetizable, contenido en un estuche cuadrado de protección. Los primeros ejemplares fueron producidos por IBM, con un diámetro de 8 pulgadas (8"). Quien le puso el nombre de "floppy" (flexible) al primer disquete parece ser que fue uno de los técnicos que tenía como cometido la comprobación del nuevo producto: al tomarlo en su mano por un lado se curvó en su totalidad y fue expontáneo el adjetivo "floppy" para el disco. Los diámetros disponibles, además de los antiguos y voluminosos de 8 pulgadas son 5 1/4, 3 y 3 1/2 pulgadas. Los dos últimos están contenidos en un receptáculo más rígido que la funda normal, por lo que el disco en su interior está más protegido. El mercado parece ser que se está orientando cada vez más hacia la utilización de estos nuevos discos flexibles, denominados también "microfloppy". Son usados, por ejemplo en los ordenadores Lisa y Macintosh.

CRT (TRC): Tubo de rayos catódicos. Términos derivados son: controlador CRT (o TRC), terminal CRT, etc.

JOYSTICK: Se trata de un potenciómetro doble situado en un soporte, giratorio en dos direcciones perpendiculares entre sí. Una palanca controla el mecanismo de rotación y permite accionar simultáneamente los dos potenciómetros, en proporciones variables según la inclinación de la propia palanca. El joystick produce a la salida dos tensiones, que pueden enviarse a las entradas correspondientes del ordenador y que son, una de ellas proporcional al desplazamiento de la palanca en la dirección "x", y la otra proporcional al desplazamiento en la dirección "y". El joystick es idóneo para ajustar la posición de un cursor en un plano x-y, como es la pantalla de gráficos de un terminal de alta resolución.

RATON: Se trata de una cajita, con uno o más pulsadores en la parte superior, conectada mediante un cable (la cola) al ordenador personal; recibe este nombre por su aspecto, semejante al de un ratoncillo (le faltan las orejas). Una bola de goma gira con componentes de rotación en x-y cuando el usuario hace deslizarse al ratón sobre una superficie plana preparada. La bola actúa sobre dos contactos giratorios perpendiculares entre sí, que generan impulsos cuya frecuencia es proporcional a la componente en la dirección x-y del movimiento del ratón. El ordenador personal, al que está conectado el ratón, puede reconstruir así el valor de la coordenada sobre la que está el cursor, tal como sucede con un joystick.

PADDLE: Paleta. Accesorio similar al joystick, pero con un solo potenciómetro, que genera una tensión de referencia única. Se utiliza mucho en los vídeo juegos.

TABLETA GRAFICA: Es una plancha plastificada bajo la cual una red especial de sensores reconoce la presencia de una punta que se mueve, dibujando, en una hoja de papel apoyada sobre la misma tableta. De este modo, se identifica la posición de la punta y se transforma en coordenadas x-y, que luego se transmiten al ordenador en forma de datos binarios. Así, el ordenador puede almacenar por puntos un dibujo mientras que el usuario lo está realizando con la tableta.



# CAPITULO VIII

## *CONCLUSIONES*

EN el momento presente, nos encontramos en una situación extremadamente confusa en lo que respecta al conocimiento y empleo de los ordenadores, tanto personales como "domésticos". Hay quienes creen haber resuelto todos sus problemas con la simple adquisición de un equipo, quizá costoso, que luego permanecerá casi siempre apagado, aunque, eso sí, dando un imponente aspecto al escritorio. Digamos la verdad: programar es todavía fatigoso y solamente las nuevas ideas de máquinas como el Macintosh comienzan a abrir amplias posibilidades de empleo del ordenador personal, incluso para quienes no tienen muchas ganas de trabajar.

Como en todas las cosas, es preciso tener (o adquirir) conocimientos y familiarizarse poco a poco con el ordenador; de no ser así, se corre el riesgo de darse cuenta "trágicamente" de que no se sabe hacer nada o casi nada con él. A pesar de ello, incluso en la actual confusión, el empleo de estas máquinas maravillosas en cualquiera de sus formas y en el ambiente que sea, siempre resulta positivo, puesto que la nueva generación estará habituada a pensar y a trabajar de un modo lógico, sirviéndose del ordenador personal en casa, en la escuela o en la oficina.

Pero, para comenzar bien, es preciso reflexionar a la hora de elegir el sistema y orientarse hacia la adquisición

de ordenadores y periféricos seguros y fiables. Es indispensable evitar caer en las "garras" de quienes especulan con la ignorancia (inicial) del comprador de su primer ordenador; por otra parte, es preciso evitar "embarcarse" en inversiones que pueden ser que no tengan sentido ni justificación en comparación con el uso que se hará de la máquina. En resumen, agenciarse un buen sistema es siempre algo difícil; es preciso asesorarse bien, sobre todo habida cuenta de las numerosas categorías y precios de cada uno de los productos existentes actualmente en el mercado, incluyendo los periféricos. Criterios más detallados que sirvan de guía en la elección de un ordenador personal serán objeto de un próximo volumen de la BBI. No obstante, antes de la adquisición, es indispensable conocer cómo funcionan, de modo que se puedan apreciar sus características más importantes y, con frecuencia, menos patentes; este era nuestro objetivo por lo que nos consideraríamos satisfechos si, después del esfuerzo que nos ha supuesto su redacción (y suponemos que al lector su comprensión), usted tuviera las ideas un poco más claras sobre lo que hay dentro y fuera de esa increible máquina que es el ordenador personal.



# BIBLIOGRAFIA

Para quienes quieran saber más. (Bibliografía esencial).


Curso técnico de introducción a la electrónica.
*Ed. Ingelek, S. A.*

Electrónica lógica y microprocesadores.
E. Santamaría. *Ed. Ingelek, S. A.*

Microprocesadores. Diseño práctico de sistemas.
J. M. Angulo. *Ed. Paraninfo.*

Los ordenadores. Fundamentos y sistemas.
J. Giarratano. *Ed. Díaz de Santos.*

6502 Assembly Language Programming
A. Leventhal. *Ed. McGraw-Hill.*

Junior Computer.
*Ed. Ingelek, S. A.*

Cibernética. Aspectos y leyendas actuales.
G.ª Santesmases. *Ed. Paraninfo.*

Construya una microcomputadora basada en el Z80.
Steve Ciarcia. *Ed. McGraw-Hill.*

Microprocesadores y microcomputadores.
Mundo Electrónico. *Ed. Marcombo.*

6502 Programming & Hardware Manual:
*Rockwell*

Z80 Programming & Hardware Manual.
*Zilog.*

# NOTAS

EN este primer volumen de Biblioteca Básica Informática se trata, fundamentalmente, todo lo relacionado con el hardware o "parte física" de un ordenador, es decir, lo que podemos tocar.

Así, términos aparentemente oscuros como CPU, microprocesador, periféricos, RAM, código ASCII, etc. y otros conceptos difíciles como la "arquitectura interna" de un ordenador o comunicación hombre-máquina son explicados con un lenguaje sencillo y desenfadado que permite asimilarlos fácilmente incluso por personas sin conocimientos previos de informática.

Por otra parte, como una ayuda más al lector, a lo largo del texto los conceptos más importantes van resaltados en azul, lo que facilita enormemente su localización en las lecturas posteriores de esta obra, que sin duda se producirán.

375 pts.